



# Design, Construction, and Qualification of a Microscale Heater Array for Use in Boiling Heat Transfer

T.D. Rule  
Washington State University, Pullman, Washington

J. Kim  
University of Denver, Denver, Colorado

T.S. Kalkur  
University of Colorado, Colorado Springs, Colorado

Prepared under Grant NAG3-1609

National Aeronautics and  
Space Administration

Lewis Research Center

## Acknowledgments

The following work was sponsored by the Microgravity Science and Applications Division of NASA under grant number NAG3-1609. The authors thank the grant monitor, Mr. John McQuillen, for his guidance and support, including acquisition of a NASA boiling chamber which we adapted for our experiment. The authors also thank Mr. Richard Quine for designing and testing the electronic control and the custom A/D system, Mr. Mark Pelletier for designing and building the computer control board, Mr. Choon Ping Ch'ng for assistance in the initial design and circuit board layout, and Mr. John Mullen for assistance with the construction of the control system.

Trade names or manufacturers' names are used in this report for identification only. This usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

### Available from

NASA Center for Aerospace Information  
7121 Standard Drive  
Hanover, MD 21076  
Price Code: A12

National Technical Information Service  
5287 Port Royal Road  
Springfield, VA 22100  
Price Code: A12

## ABSTRACT

Boiling heat transfer is an efficient means of heat transfer because a large amount of heat can be removed from a surface using a relatively small temperature difference between the surface and the bulk liquid. However, the mechanisms that govern boiling heat transfer are not well understood. Measurements of wall temperature and heat flux near the wall would add to the database of knowledge which is necessary to understand the mechanisms of nucleate boiling.

A heater array has been developed which contains 96 heater elements within a 2.5 mm square area. The temperature of each heater element is held constant by an electronic control system similar to a hot-wire anemometer. The voltage that is being applied to each heater element can be measured and digitized using a high-speed A/D converter, and this digital information can be compiled into a series of heat-flux maps. Information for up to 10,000 heat flux maps can be obtained each second.

The heater control system, the A/D system and the heater array construction are described in detail. Results are presented which show that this is an effective method of measuring the local heat flux during nucleate and transition boiling. Heat flux maps are obtained for pool boiling in FC-72 on a horizontal surface. Local heat flux variations are shown to be three to six times larger than variations in the spatially averaged heat flux.

## TABLE OF CONTENTS

Chapter 1: Introduction .....	1
1.1. Background .....	1
1.2. Research Objectives .....	3
1.2.1. Constant Temperature Boundary Condition .....	6
1.3. Summary of Completed Work .....	6
1.4. Figures .....	7
Chapter 2: Heater Construction.....	8
2.1. History of Heater Design.....	8
2.1.1. Silicon Substrate, SiO <sub>2</sub> insulating layer .....	9
2.1.2. Quartz substrate, SiO <sub>2</sub> layer .....	11
2.1.3. Present Design.....	13
2.1.4. Heater Specifications.....	16
2.2. Figures .....	17
Chapter 3: Constant Temperature Control System and data acquisition .....	21
3.1. Feedback Board.....	21
3.1.1. Voltage-Controlled Resistance.....	22
3.1.2. Transient Response .....	24
3.2. Computer Control Board.....	25
3.3. A/D Systems.....	29
3.3.1. IOtech A/D system.....	29
3.3.2. High-speed A/D board .....	30
3.4. Physical Arrangement of Control Hardware.....	32
3.4.1. Backplane .....	32



3.4.2. Decoding board .....	33
3.5. Ground null circuitry .....	34
3.6. Calibration Method .....	37
3.7. Trimming.....	38
3.7.1. Ground potential variations across the motherboard.....	43
3.8. Software .....	44
3.9. Figures .....	48
Chapter 4: Experimental Apparatus .....	59
4.1. Calibration Apparatus .....	59
4.2. Calibration method.....	60
4.3. Test Chamber .....	61
4.4. Figures .....	64
Chapter 5: Heat Transfer Behavior on Small Horizontal Heaters During Pool Boiling of FC-72 .....	67
5.1. Experimental Procedure .....	67
5.1.1. Degassing of fluid .....	67
5.1.2. Heater calibration .....	68
5.1.3. Data acquisition.....	68
5.1.4. Data reduction .....	69
5.1.5. Uncertainty analysis .....	70
5.2. Results .....	71
5.2.1. Preliminary space-resolved, time-resolved results.....	71
5.2.2. Spatially averaged, time averaged boiling curve .....	73

5.2.3. Spatially resolved, time averaged RMS .....	74
5.2.4. Spatially averaged, time resolved results .....	76
5.2.5. Time resolved data from individual heaters .....	77
5.2.6. Spatially resolved, time averaged data .....	78
5.2.7. Conditional sampling .....	79
5.2.8. Boiling fraction .....	80
5.2.9. Conditionally sampled heat flux .....	81
5.3. conclusions .....	82
5.4. Figures .....	83
References .....	97
Appendix A: Derivation of Offset Error .....	100
A.1. Derivation of Equivalent Resistance .....	100
A.2. Derivation of Offset Errors.....	101
A.3. Effect of trimming on offset errors.....	102
Appendix B: Software Operating Instructions .....	106
B.1. CAL .....	106
B.1.1. Description of Controls in "Heater Array Calibration" Window .....	109
B.1.2. Description of Controls in "Heater Calibration Setup" Window .....	110
B.2. CONTROL .....	112
B.2.1. "Control" Window Controls .....	116
B.2.2. "Data Acquisition Setup" Window .....	117
B.3. Post-Processing.....	118
B.3.1. "Post-Processing" Window Controls.....	122

B.4. GRAPH.....	125
B.4.1. "Graph" Window Controls .....	126
B.5. File Format Specifications .....	127
B.5.1. Calibration File Format .....	127
B.5.2. Binary Data File Format .....	128
B.5.3. Control Setup File Format .....	128
B.5.4. Tag File Format .....	129
B.5.5. Resistance File Format .....	130
B.5.6. Offset File Format .....	130
B.5.7. Size File Format.....	130
B.6. Figures .....	131
Appendix C: Custom A/D System .....	136
C.1. Connections and I/O operations .....	136
C.1.1. Cabling.....	136
C.1.2. Data acquisition and downloading procedure .....	137
C.2. Details of A/D Board Design.....	139
C.2.1. Introduction .....	139
C.2.2. Analog to Digital Converter Chip Control Signals .....	142
C.2.3. Multiplexer Control Signals .....	143
C.2.4. Memory Control Signals .....	143
C.2.5. Conditioning Analog Circuits.....	145
Appendix D: Connections Between Components .....	146
Appendix E: Datasheets for key parts .....	149

Appendix F: Control System Schematics.....	156
F.1. Feedback Control Board Schematic.....	157
F.2. Microprocessor Control Board Schematic.....	159
F.3. Custom High-Speed A/D Board .....	162
F.4. Motherboard Schematic .....	166
Appendix G: Source Code Listings for Visual Basic Programs .....	173
G.1. Listing of “CONTROL.VBP” .....	173
G.1.1. Listing of MODADC.BAS.....	173
G.1.2. Listing of MODSHARE.BAS .....	183
G.1.3. Listing of MODCONVE.BAS .....	186
G.1.4. Listing of FRMAUTOM.FRM.....	207
G.1.5. Listing of FRMCONFI.FRM .....	213
G.1.6. Listing of FRMVIEWC.FRM .....	213
G.1.7. Listing of FRMDAC.FRM.....	213
G.1.8. Listing of FRMADC.FRM.....	214
G.1.9. Listing of FRMAUTO.FRM .....	214
G.1.10. Listing of FRMBATCH.FRM.....	218
G.2. Listing of “CAL.VBP” .....	222
G.2.1. Listing of MODDAC.BAS.....	222
G.2.2. Listing of MODMAIN.BAS.....	227
G.2.3. Listing of FRMSETUP.FRM .....	227
G.2.4. Listing of FRMMAIN.FRM.....	234
G.3. Listing of GRAPH1.VBP .....	241

G.3.1. Listing of MODGRPAH.BAS.....	241
G.3.2. Listing of FRMMAIN.FRM.....	242

## LIST OF FIGURES

Figure 1.1: Schematic of heat conduction between heaters of different temperature .....	7
Figure 2.1: Photograph of serpentine platinum resistance heater .....	17
Figure 2.2: Cross-section view of construction of first-generation heater .....	17
Figure 2.3: Cross-section view of construction of second-generation heater .....	17
Figure 2.4: Power leads routed between serpentine heaters near edge of heater array .....	18
Figure 2.5: Cross-section view of construction of third-generation heater .....	18
Figure 2.6: Heater arrangement of third-generation heater .....	19
Figure 2.7: Schematic of quartz wafer mounted on the PGA package .....	19
Figure 2.8: Wire bonding method .....	20
Figure 3.1: Schematic of control and data acquisition system components .....	48
Figure 3.2: Functional schematic of feedback control circuit .....	48
Figure 3.3: Analog multiplier configured as a voltage-controlled resistance .....	49
Figure 3.4: Complete schematic of feedback control circuit. ....	49
Figure 3.5: Transient response of feedback circuit .....	50
Figure 3.6: Computer control board multiplexing scheme .....	50
Figure 3.7: Timing for multiplexing method .....	51
Figure 3.8: Daqbook Multiplexing Scheme .....	51
Figure 3.9: Arrangement of backplane connectors - Front View .....	52
Figure 3.10: Arrangement of backplane connectors - Back View .....	52
Figure 3.11: Decoding card layout .....	53
Figure 3.12: Printed circuit board to hold heater array .....	53
Figure 3.13: Common ground impedance .....	54

Figure 3.14: Effect of ground impedance in feedback circuit.....	54
Figure 3.15: Effect of ground impedance in feedback circuit.....	55
Figure 3.16: Effect of ground impedance in feedback circuit.....	55
Figure 3.17: Schematic of ground potential experiment.....	56
Figure 3.18: Schematic of ground null circuit.....	56
Figure 3.19: Ideal control-circuit response for constant-resistance .....	56
Figure 3.20: Heater Voltage vs. Control Voltage, Constant Temperature Lines. ....	57
Figure 3.21: Circuit for correcting analog multiplier output offset.....	57
Figure 3.22: Alternative circuit for correcting analog multiplier output offset .....	57
Figure 3.23: Modification to Figure 3.22 for correcting analog multiplier output offset .....	58
Figure 3.24: Ground voltage at various points on backplane board. Measurements are given in units of mV.....	58
Figure 4.1: Calibration Apparatus.....	64
Figure 4.2: Heater temperature rise with and without impinging jet .....	65
Figure 4.3: Heater temperature rise showing threshold voltage.....	65
Figure 4.4: Experimental Apparatus .....	66
Figure 4.5: Temperature indicator calibration.....	66
Figure 5.1: Changes in array averaged heat flux over time at $\Delta T_{\text{sat}} = 17.5\text{ }^{\circ}\text{C}$ . ....	83
Figure 5.2: Uncorrected boiling curve and substrate conduction data. ....	84
Figure 5.3: Arrangement of 96 heaters in the array, with non-functional heaters represented as black squares.....	84
Figure 5.4: Variation in heat transfer coefficient vs. time for a single heater.....	85
Figure 5.5: Space-averaged heat transfer coefficient vs. Time .....	85

Figure 5.6: Surface plots of the local heat transfer coefficient over 64 center heaters (1 major tick mark in the Z-direction = $5500 \text{ W/m}^2\text{-K}$ , major tick mark closest to horizontal axes = $0 \text{ W/m}^2\text{-K}$ ) .....	86
Figure 5.7: Arrangement of 96 heaters in the array, with non-functional heaters represented as black squares.....	87
Figure 5.8: Schematic of experimental apparatus. ....	88
Figure 5.9: Boiling curve showing RMS variation range and uncertainty bars. ....	88
Figure 5.10: Spatially resolved vs. spatially averaged RMS heat flux variation. ....	89
Figure 5.11: Spatially resolved average and RMS variations in heat flux at three superheats .....	90
Figure 5.12: Heat flux vs. Time for a heater in the center of the array at $\Delta T_{\text{sat}}=47.5 \text{ }^\circ\text{C}$ (transition boiling).....	91
Figure 5.13: Array averaged, time resolved heat flux vs. time. ....	91
Figure 5.14: Heat flux vs. time for heater #18 along with the boiling function.....	92
Figure 5.15: Boiling curves for "rings" of heaters .....	95
Figure 5.16: Schematic of boiling function generation method.....	95
Figure 5.17: Boiling fraction vs. wall superheat for "rings" of heaters.....	95
Figure 5.18: Boiling heat flux for "rings" of heaters.....	96
Figure A.1: Equivalent resistance circuit using multiplier chip.....	105
Figure B.1: CAL window .....	131
Figure B.2: CAL setup window .....	132
Figure B.3: CONTROL window .....	132
Figure B.4: CONTROL setup window .....	133



Figure B.5: CONTROL automation window .....	134
Figure B.6: CONTROL post-processing window .....	135
Figure B.7: GRAPH window .....	135
Figure C.1: Timing Diagram of A/D System .....	142



## **CHAPTER 1: INTRODUCTION**

### **1.1. BACKGROUND**

Boiling is an efficient means of removing heat from a surface due to the large amounts of heat that can be removed with a relatively small temperature difference between the wall and the fluid. However, the mechanisms that govern heat transfer in boiling are not well understood, because of the difficulty of making direct measurements of quantities such as temperature, heat flux and wetted area at the wall during nucleation, growth and departure. Direct measurements of these quantities at the wall beneath a growing and departing bubble are important for understanding the relative contribution of various mechanisms to the overall heat transfer in boiling. In spite of much progress toward measuring these quantities in greater detail, most studies refer to single-point or time- and space-averaged wall temperature and heat flux measurements.

At critical heat flux and during transition boiling, heat transfer is due to vapor-liquid structures that are more complicated than the structure of a single bubble. For these cases, measurement of physical quantities at the wall will provide a better understanding of what these structures are and how they interact to result in a critical heat flux condition.

Much progress has been made toward making better surface temperature and heat flux measurements. Cooper and Lloyd (1969) used a microthermocouple to measure temperature fluctuations beneath a growing bubble, and demonstrated the existence of the microlayer. Some more recent works measured the rate of change of the microlayer thickness, inferring from this the local heat flux (Fath and Judd, 1978, Judd and Hwang, 1976, Koffman and Plesset, 1983, Shoukri and Judd, 1975, and Voutsinos and Judd, 1975). Kenning (1992) and Watwe and

Hollingsworth (1994) obtained information about spatial temperature variations using liquid crystals on a thin stainless steel sheet, but this data had limited temporal resolution.

Several numerical investigations emphasize the need to make detailed temperature and heat-flux measurements. Lee and Nydahl (1989) showed that microlayer evaporation accounted for 87 percent of the heat transfer enhancement in pool boiling for a particular case, but they lacked experimental data with which to validate their model. Unal and Pasamehmetoglu (1994) showed that spatial and temporal surface temperature variations have significant effects on spatial and temporal heat flux, and that microthermocouples buried near a heating surface may not accurately measure the heater surface temperature.

The vast majority of experimental work performed to date regarding boiling has utilized single heaters that were large compared to individual bubble sizes, making it difficult to look at details of the boiling process. These experiments usually used a heating element operated in a constant heat flux mode, making it difficult to study transition boiling effects beyond critical heat flux (CHF). Other experiments have utilized surfaces held at constant temperature, but the *local* heat flux and temperature were not measurable and can vary significantly across the heater. Even when local measurements were obtained (e.g., Cooper and Lloyd, 1969, Lee, et al., 1985, Marquardt and Auracher, 1990, Hohl, et al., 1997), this was done at only a few locations on the heater surface.

The pioneering work of Kenning (1992) and Watwe and Hollingsworth (1994) using liquid crystals on thin, electrically heated stainless steel plates did much to elucidate the heat transfer mechanisms associated with large scale phenomenon (e.g. the role of bubble driven convective flows, the spread of boiling on large scale heaters, and nucleation site interactions) because information regarding temperature fluctuations were available with high resolution

across the heater surface. The work described in this paper complements the liquid crystal work in that boiling on a comparatively small heated area is investigated in detail with high spatial and temporal resolution. Also, this work was performed with a constant wall temperature instead of a constant wall heat flux boundary condition, simulating boiling on a thick surface with high thermal conductivity.

The behavior of boiling on small heated areas can differ from that on large heated areas. First, the total number of nucleation sites is much smaller, and can result in heaters smaller than the corresponding average distance between nucleation sites on large heaters. Boiling can be delayed to higher wall superheats as a result, or the number of nucleation sites may not be statistically representative. Second, the Taylor wavelength, which is significant in transition and film boiling, can be larger than the heater size, altering boiling behavior in these regions. Third, edge-effects can become significant.

## **1.2. RESEARCH OBJECTIVES**

The motivation of this research was the lack of detailed information about the wall heat flux and wall temperature during nucleate and transition boiling. There is a need for a method of measuring the heat flux and temperature at many points on the wall with high spatial and temporal resolution, so that the mechanisms of heat transfer at various points on the boiling curve can be better understood.

The following specific research objectives arose out of the need for more detailed surface heat transfer information.

1. A heater array would be developed which would provide a map of both the temperature and the heat flux on a heated surface with high spatial and temporal resolution.

2. Boiling curves for a small square heated area would be obtained, including critical heat flux and the transition boiling regimes.
3. Spatial and temporal variations in the wall heat flux over the heater array would be measured at various points on the boiling curve.
4. The mechanisms of nucleate boiling, transition boiling and critical heat flux would be clarified.

Both heat flux and temperature measurements are needed because it has been shown in other studies that both of these quantities vary significantly in space and time during nucleate boiling, and that these variations must be considered in physical models of the boiling process. An array of constant-temperature heater elements was used to accomplish these objectives. The heater array and its associated control and data acquisition electronics allow the temperature of the surface to be controlled and the power dissipation from each heater to be measured. There are several advantages to using a constant-temperature heater system. These advantages are explained in detail.

1. The conduction through the substrate between adjacent heaters is minimized.
2. The need to solve a transient heat conduction problem to obtain the heat flux through the surface is eliminated.
3. A constant-temperature boundary condition is achieved, which simplifies the analytical and numerical solution of the system.
4. Data can easily be obtained in the critical heat flux and transition boiling regions without the danger of heater dryout.

One disadvantage of the constant temperature system is the cost and complication of the constant-temperature control system.

Figure 1.1 shows how a temperature difference between two adjacent heaters affects the heat transfer measurement. The term  $q_h$  represents the heat generation per unit area of the heater element on the left. This quantity can be measured by sensing the voltage across the heater and the resistance of the heater. The term  $q_{liq}$  represents the amount of heat that is transferred into the liquid above the heater,  $q_{sub}$  is the heat flux into the adjacent heaters due to local temperature differences. The term  $q_{steady}$  represents the steady state heat conduction into the surrounding substrate due to convection from the unheated substrate surface. The total heat balance on the heater is:

$$q_h = q_{liq} + q_{steady} + q_{sub}$$

$q_{steady}$  can be measured directly for a given average surface temperature value. If there is a significant temperature difference between two adjacent heaters due to heat transfer on the surface, such that  $T_1 - T_2 = \Delta T$ , then a significant amount of heat will be conducted through the substrate from one heater to the other, especially since the length scales are small. The result is that the measured heat generation from the heater area,  $q_h$ , will not reflect the heat flux into the liquid,  $q_{liq}$ , because there is no way to directly measure  $q_{sub}$ .

There are two solutions to the problem of quantifying  $q_{sub}$ . One solution is to measure both the temperature and the heat flux at each surface heater with high spatial and temporal resolution. A numerical solution to a transient heat conduction problem can be performed using the results, in order to calculate the transient substrate conduction numerically. The other solution is to maintain each individual heater at a constant temperature, so that  $T_1 = T_2$  in Figure 1.1. If the temperature difference between heaters is zero, then the heat flux between the heaters will also be zero, and the  $q_{sub}$  term is eliminated.

### **1.2.1. Constant Temperature Boundary Condition**

In boiling models, the variations in both heat flux and temperature on the surface due to conduction in the substrate can be important. By eliminating conduction in the substrate and providing a constant-temperature boundary condition, numerical or analytical models of nucleate and transition boiling, and critical heat flux, can be simplified. Once models of this simplified boiling problem are demonstrated to be effective, the transient conduction portion of the problem can be added to the model.

## **1.3. SUMMARY OF COMPLETED WORK**

In order to meet the objectives, a microscale heater array was designed that is capable of measuring both the surface temperature and heat flux with much higher spatial and temporal resolution than in previous studies. This is achieved using an array of 96 independently controlled heaters operated in a constant temperature mode. Feedback loop circuits similar to those used in hot-wire anemometry are used to keep each heater at a constant temperature, and the power required to do this is measured, enabling the heat transfer coefficient from each heater to be determined. Such information can provide much needed data regarding the important heat transfer mechanisms during the bubble departure cycle, and can serve as a benchmark to validate many of the analytical and numerical codes used to model boiling. The experiment design and results are presented in the following chapters.

The information that is obtained using this method is unique in that data is taken at many points simultaneously instead of at a single point, enabling a much more detailed picture of the heat transfer process to be obtained. An additional advantage of this work is that the heat flux is measured directly, instead of being inferred from average heat flux data and void fraction measurements.



Chapter 2 describes the construction of the microscale heater array that was used in the experiment.

Chapter 3 describes the principles of operation of the electronic control system and data acquisition system.

Chapter 4 describes the experimental procedure, including the calibration method and the experimental apparatus.

Chapter 5 discusses the magnitude of local heat flux variations with time, and shows that the local heat flux variations are significant and are not represented by the space-averaged heat flux which is typically measured, and which is often used in physical models. It examines the relationship of heat flux to the fraction of the surface that sees boiling activity. It also summarizes the uncertainties in the temperature measurements.

#### 1.4. FIGURES

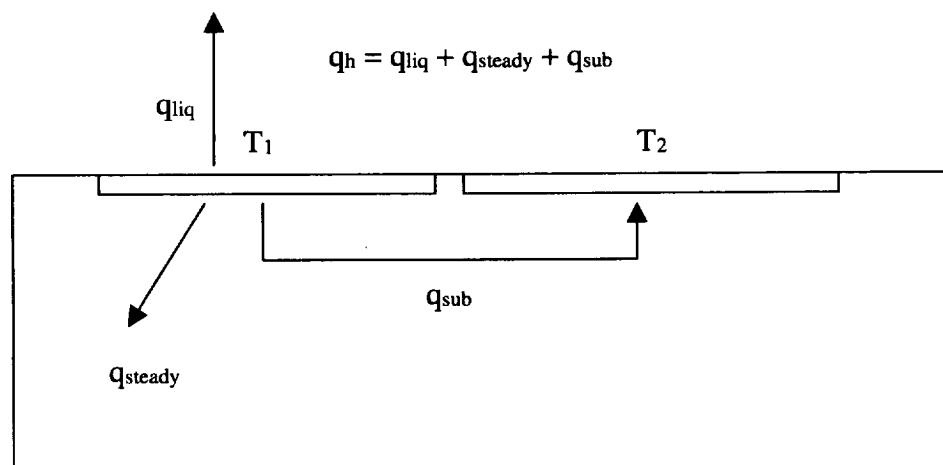


Figure 1.1: Schematic of heat conduction between heaters of different temperature

## **CHAPTER 2: HEATER CONSTRUCTION**

This chapter describes in detail the physical arrangement and construction of a microscale heater array that is used to make high-resolution heat flux measurements on a constant-temperature surface. This array is constructed by depositing and etching away layers of conductive and insulating material on a substrate to form conductive paths on the surface which will dissipate heat when electrical current is passed through them. The heaters are maintained at a constant temperature by an electronic control system, which is described in Chapter 3.

### **2.1. HISTORY OF HEATER DESIGN**

Several generations of heater arrays were built and tested before a design was constructed that provided acceptable performance. The first generation used a highly doped silicon wafer as a substrate. The bulk wafer served as a common ground for all the heater elements. Problems with the insulating layer between the heaters and the power leads and concerns about substrate heat conduction led to a second generation array, which used a quartz wafer as a substrate, and a plane of aluminum to serve as a common ground for all the heater elements. Problems with large voltage drops through the aluminum ground layer led to the present generation of heater array, which uses a quartz substrate with an individual ground wire for each heater element.

The basic element of the microscale heater arrays is the serpentine platinum heater, shown in Figure 2.1. The bright areas in the figure are the platinum heater lines. This heater element is constructed by depositing platinum onto the substrate surface, masking off the heater lines, and etching the platinum away from the unmasked areas. The terminal ends of each platinum heater are connected to the edge of the chip with aluminum leads deposited on the chip.

Photomasks are used to expose a layer of photoresist that is deposited onto the wafer that is being processed. The photoresist can be either chemically removed or chemically fixed in place wherever the photomask allows ultraviolet light to pass through. The photoresist then serves to mask off portions of the wafer from processes such as wet chemical etching or ion milling. Each mask was designed using the L-EDIT program for VLSI design, running on an Apple Macintosh computer.

Because the heater design continues to evolve, a set of photomasks is generally used to process only a few wafers. Since a set of photomasks tends to be more expensive than the resources required to process 3 or 4 additional wafers, each photomask is designed to perform one masking operation using one half of the mask, and a different masking operation using the other half. Between operations, the mask is rotated 180 degrees, so that only half of the wafer being processed is exposed to the necessary mask. As a result, only half of each processed wafer contains usable components, because half the wafer is processed using the wrong mask for each process.

### **2.1.1. Silicon substrate, Silicon Dioxide insulating layer**

The first heater array that was tested consisted of 148 rectangular platinum heaters arranged within an approximately 3 mm circle. Each heater measures  $250\text{ }\mu\text{m} \times 250\text{ }\mu\text{m}$ . The platinum heaters are separated from the doped silicon ground layer and the aluminum power lead layer by silicon dioxide layers. Holes etched in the oxide layers, called vias, provide a current path through the oxide layers to connect the heater to the power and ground connections.

When this heater was tested, it did not perform as well as it was hoped. Difficulties included a problem with conduction through pinholes in the deposited  $\text{SiO}_2$  layer, corrosion of

the aluminum power leads that were deposited on the top layer of the chip, and a high rate of thermal conduction within the silicon substrate.

One problem arises in the process of depositing the  $\text{SiO}_2$  layer. The oxide fails to deposit on certain areas of the chip, which causes microscopic "pinholes" to develop in the oxide layer. When these holes are filled with conducting material in subsequent deposition steps, they can create a short circuit between a midpoint of the platinum heater and an aluminum power lead that passes over that heater. As a result, one "pinhole" can render two heaters effectively useless: both the heater with the midpoint connection and the heater that any shorted power lead would normally supply are affected.

Another potential problem with a silicon substrate is its high thermal conductivity. It has been found in tests and analysis that the amount of conduction from the heater to the surrounding substrate is significant. It is desirable to eliminate conduction through the substrate so the power going into the heaters can be related to the heat flux into the liquid covering the heater. If this conduction is not eliminated or minimized, heat conducts from the heaters into the surrounding substrate, where the heat transfer coefficient is not known. This heat transfer cannot be quantified, and the actual heat transfer into the fluid cannot be known. This issue will be discussed in more detail in Chapter 6 under the uncertainty analysis section. This conduction will be a greater problem with a silicon substrate than with a quartz or Kapton substrate, which both have much lower thermal conductivity.

A modification of this method was attempted using silicon nitride as an insulating layer instead of  $\text{SiO}_2$ . However, the nitride layer tended to crack and peel after it was deposited, and this method was abandoned.

Acetone was initially chosen as a test fluid when experiments were first carried out to measure the performance of the microscale heaters. When the heater was tested in acetone, it was found that even low heater currents caused the aluminum power leads to deteriorate and fail. However, the leads did not fail when the heater was tested in FC-72, a dielectric fluorocarbon coolant. Acetone passes a small amount of electrical current, while FC-72 registers an essentially infinite resistance on laboratory multimeters. It is speculated that the small amount of current that acetone carries may cause an electrolytic reaction on the aluminum leads. This has not been confirmed, however.

### **2.1.2. Quartz substrate, Silicon Dioxide layer**

A new heater layout was designed in an attempt to overcome these difficulties. The problems with pinholes, aluminum corrosion and substrate conduction were addressed with changes in the heater design. Figure 2.3 shows a cross sectional view of the second-generation heater design.

Substrate conduction was reduced by using a quartz substrate instead of a silicon substrate. Silicon is 90 times more thermally conductive than quartz. Quartz is an electrical insulator, so the substrate cannot be used as an electrical ground as it was when silicon was used. A Kapton substrate would further reduce substrate conduction, but it has not yet been tested as a substrate material. Table 2.1 gives the properties of the three different substrate materials that were considered.

Material	Thermal Conductivity (W/m-K)
Silicon	135
Quartz	1.5
Kapton	0.2

Table 2.1: Properties of Substrate Materials

The pinhole problem was addressed by several changes. The first change was to route the power leads between the individual heaters to the edge of the heater array, where aluminum leads connect the heater array to the edge of the chip without passing over the top of any heaters. Routing the power leads between the heaters requires wider spacing between the heaters near the edge of the array. If the array contained 148 heaters, these gaps would be too large, so a smaller array of 112 heaters was used instead. Figure 2.4 shows how the power leads are routed between the heaters near the edge of the array.

Because the substrate was changed to an electrical insulator, an aluminum layer was deposited over the platinum heaters to serve as a common heater ground. Amorphous silicon was deposited between the heater layer and the aluminum ground layer. It was thought that the amorphous silicon would not form pinholes like silicon dioxide. Unfortunately the amorphous silicon layer peeled when it was deposited, so another method was sought to isolate the aluminum ground layer from the platinum heaters.

The higher melting point of the quartz substrate may eventually be used to eliminate the pinhole problem. Because of the higher melting point, the substrate can be heated to a high enough temperature to cause the oxide layer between the heaters and the top ground layer to flow. The pinholes may be filled in by flowing oxide, especially if multiple layers of oxide are applied, one on top of the other. A different method of cleaning the surface prior to depositing the oxide layer may also help eliminate the pinhole problem. However, no method has yet proven satisfactory, and the only solution to this problem has been to place all electrical current paths on the same layer.

The feasibility of the silicon substrate heaters was being tested while the quartz substrate heaters were being developed. During these tests, it was found that the current from a group of

heaters passing through the doped silicon ground layer caused the ground potential to vary up to 2 mV across the ground layer. These variations can cause a significant error in the temperature response of the feedback control circuit. This ground potential characteristic is described in more detail in Section 3.5

It was estimated that a similar variation in ground potential would be seen for the second-generation heater design. The thickness of the aluminum ground layer was known, and a typical current flow through the ground layer could be estimated. These quantities were used to predict a maximum possible ground voltage potential.

### **2.1.3. Present Design**

The simplest way to eliminate the problem with ground potential variations is to provide an individual ground lead for each heater which connects to a ground bus bar. This bus bar must be large enough to provide less than 100  $\mu\text{V}$  voltage difference between individual heater ground connections during all operating conditions. A new heater layout was created which provides an individual power and ground lead for each heater. The heaters and the power leads were deposited on the same layer, to simplify construction. A cross section of the heater construction is shown in Figure 2.5, and the arrangement of the heaters in the array is illustrated in Figure 2.6.

This heater array is constructed using a single conductive layer. Heater construction is simplified because no vias need to be etched. The following steps are followed in its fabrication.

1. A thin layer of titanium is first sputtered onto the quartz to enable the platinum to adhere to the surface.
2. A 2000 Å layer of platinum is deposited on top of the titanium layer.
3. The platinum and titanium are etched away to leave the serpentine platinum heaters and the power leads.

4. A layer of aluminum is then deposited and etched away to leave aluminum overlapping the platinum for the power leads and the wire bonding pads.
5. Finally, a layer of silicon dioxide is deposited over the heater array to provide a uniform surface energy across the heater. The area where wire-bond connections will later be made is masked off to maintain a bare aluminum surface.

Once the quartz wafer containing multiple heater arrays is prepared, the wafer is diced into smaller square sections, each containing a single heater array. Dicing is performed with an excimer laser or a diamond saw. The individual heater arrays are each mounted onto a pin grid array package (PGA). The PGA is plugged into a socket so that heater arrays can be easily replaced.

The present heater design consists of a pattern of heater lines and power leads on a transparent substrate. The heater lines are spaced one line-width apart, so that they only partially obscure vision through the heater array. Thus, boiling activity on the surface of the heater can be visually observed from the back of the substrate. This will allow boiling events that occur near the surface to be visually observed from behind the substrate.

Several methods were considered for creating a preferential nucleation site at the center of the heater array so that the position of a single growing bubble on the heater could be fixed at the most desirable location. The preferred site dimensions would be approximately 1 micron in diameter and as deep as possible. A re-entrant cavity would be ideal, in where the mouth of the cavity is smaller in diameter than the interior of the cavity. These cavities tend to trap vapor or dissolved gas within the cavity because surface tension forces cause a very low pressure within the cavity, which prevents dissolved gas from re-dissolving in the solution or vapor from condensing.



A CO<sub>2</sub> laser was tested for drilling this hole, but the quartz did not absorb enough of the laser energy to allow such a small hole to be cut. It may be possible to create such a site using an excimer laser, but the cost of this manufacturing process is prohibitive. A chemical etching method may be practical, but the hole formed by this method would probably be very shallow compared to its diameter.

The quartz heater array substrate is mounted to the PGA package as shown in Figure 2.7. The PGA package is first drilled with a hole so that the chip can be viewed from the back. Transparent epoxy adhesive is used to bond the heater array substrate to the PGA package, so that the heater can be viewed from the back of the substrate. A glass coverslip on the back of the PGA package is used to provide an optically flat surface on the back of the PGA, and to close the cavity that contains the epoxy adhesive.

Once the heater array substrate is attached to the PGA, electrical connections must be made between the PGA and the heater array. This is accomplished using a wire bonding method. Both the PGA and the heater array have pads to which wire bonds can be made.

The wire bonding technique that was used is illustrated in Figure 2.8. Figure 2.8a shows the wire capillary tip with the gold wire and a gold ball formed at the end of the wire. In Figure 2.8b, this gold ball is then pressed down on the workpiece where the first connection is to be made. Pressure and an ultrasonic pulse cause a metallurgical bond to form between the gold ball and the workpiece. In Figure 2.8c and Figure 2.8d, the tip is drawn up and repositioned over the site of the second bond. In Figure 2.8e pressure and an ultrasonic pulse cause another bond to form, called a wedge bond. The wire is pinched where the tip presses against the surface, and when the tip is lifted in Figure 2.8f, the wire breaks off. Finally, in Figure 2.8g, an electric spark

from the electrode shown below the tip causes another gold ball to form, and another wire bond can be made by repeating steps a through g.

It was very difficult to make the wire-bonding connection to the aluminum power and ground leads on the heater array. Either the bonds would not adhere to the aluminum, or else they adhered and pulled the aluminum off of the substrate when the ball-bonder tip was raised. The ball-bonds that did adhere to the aluminum were weak and unreliable. These problems may be caused by weak adhesion between the aluminum and the quartz substrate.

#### **2.1.4. Heater Specifications**

The finished heater array measures approximately 2.7 mm square. Each individual heater, as shown in Figure 2.1, is approximately 0.27 mm square. The lines of the serpentine pattern are 5  $\mu\text{m}$  wide, with 5  $\mu\text{m}$  spaces in between the lines. The total length of the platinum lines in one heater is about 6000  $\mu\text{m}$ , and the heater lines are about 2000 Å thick. These dimensions result in an overall heater resistance of approximately 1000  $\Omega$ . The change in resistance with temperature is on the order of  $2 \times 10^{-3} \Omega/\Omega\text{-}^\circ\text{C}$ , resulting in a change of about 2  $\Omega/^\circ\text{C}$  for the given heater dimensions. Current flowing through the heater causes electrical power to be dissipated as thermal heat, and the change in resistance with temperature allows the temperature to be determined by measuring the resistance of the heater. The total resistance and resistance/temperature relationship are high enough that contact resistance in the connections will cause negligible errors in the temperature measurement. However, the resistance must also be low enough that the voltages required to power the heaters will not be too high for typical off-the-shelf parts to be used in the control system electronics.

## 2.2. FIGURES

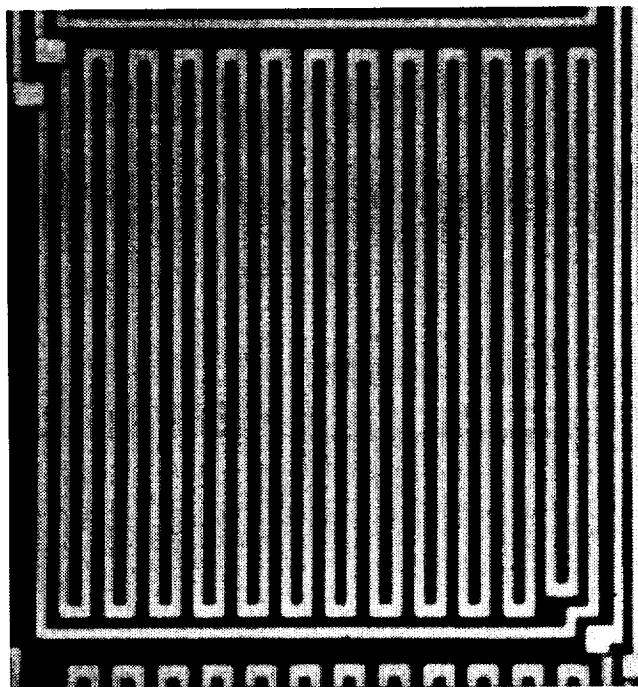


Figure 2.1: Photograph of serpentine platinum resistance heater

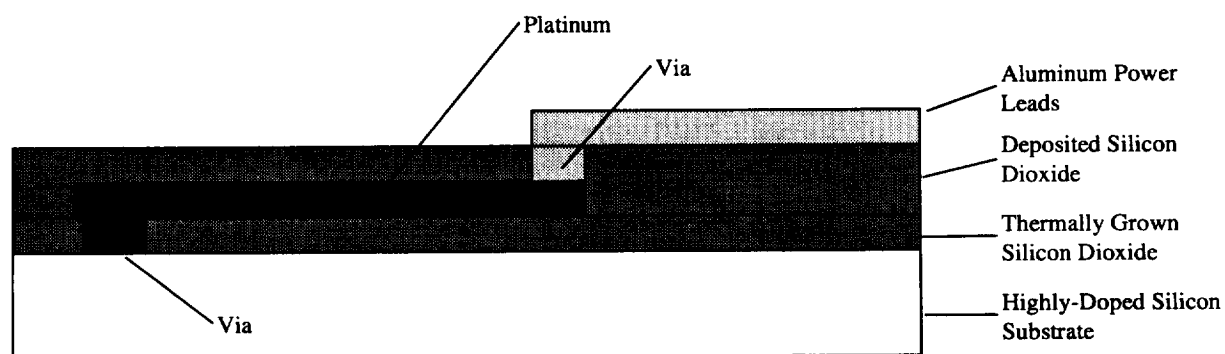


Figure 2.2: Cross-section view of construction of first-generation heater

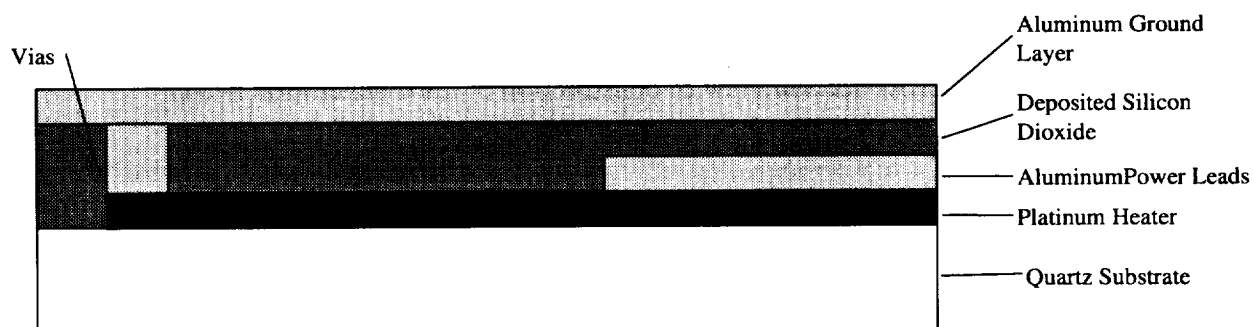


Figure 2.3: Cross-section view of construction of second-generation heater

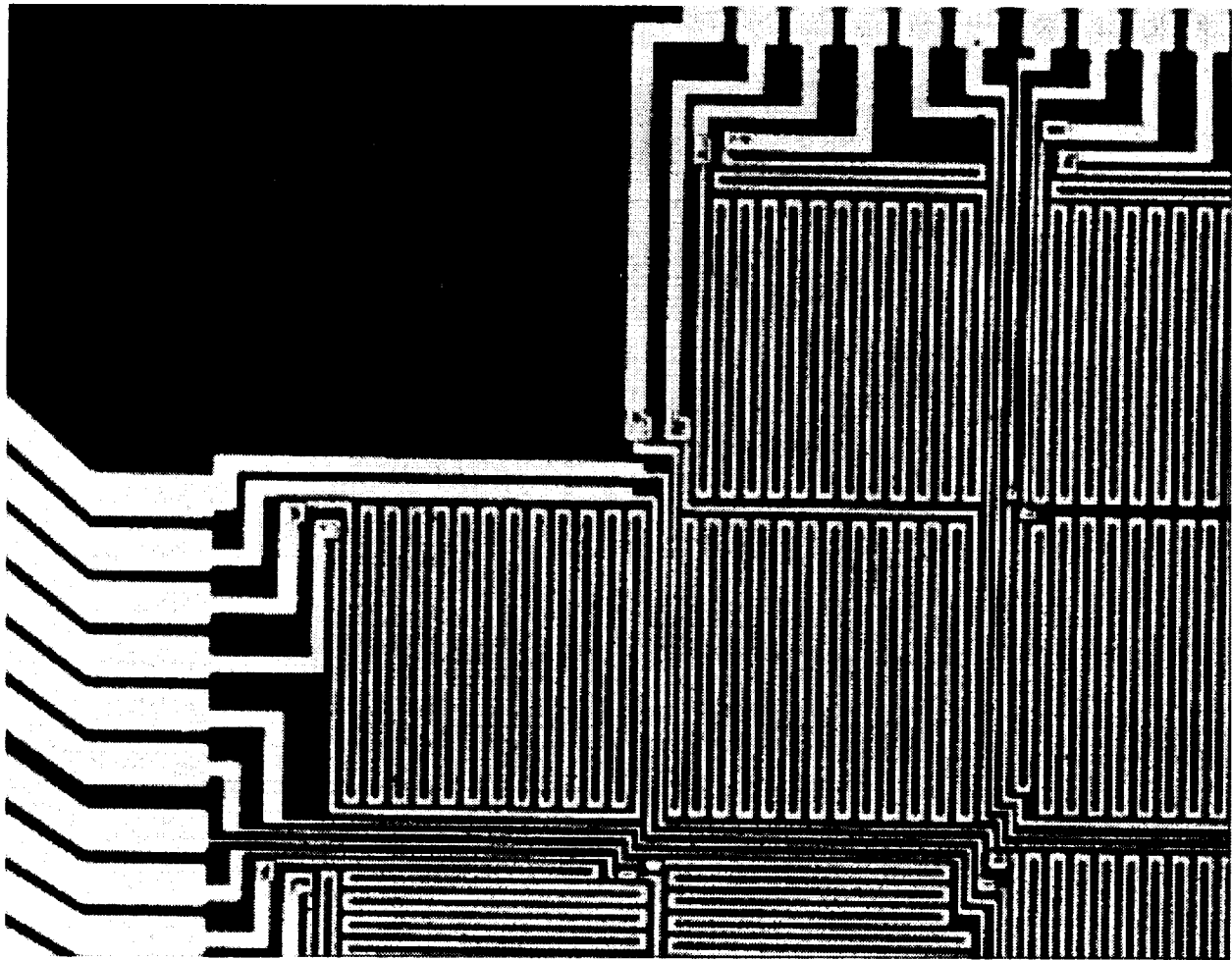


Figure 2.4: Power leads routed between serpentine heaters near edge of heater array

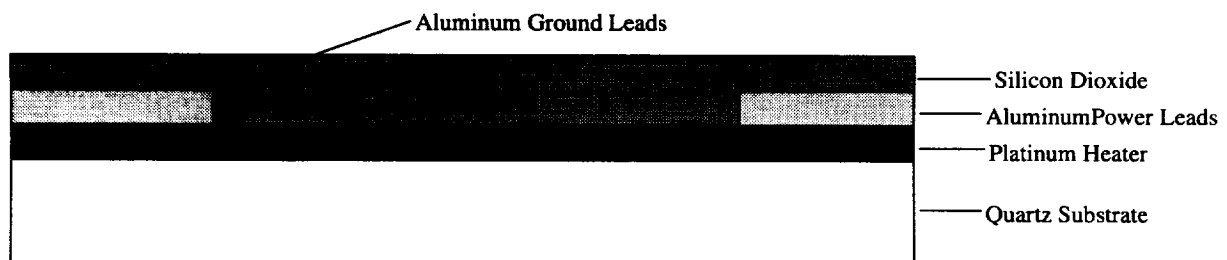


Figure 2.5: Cross-section view of construction of third-generation heater

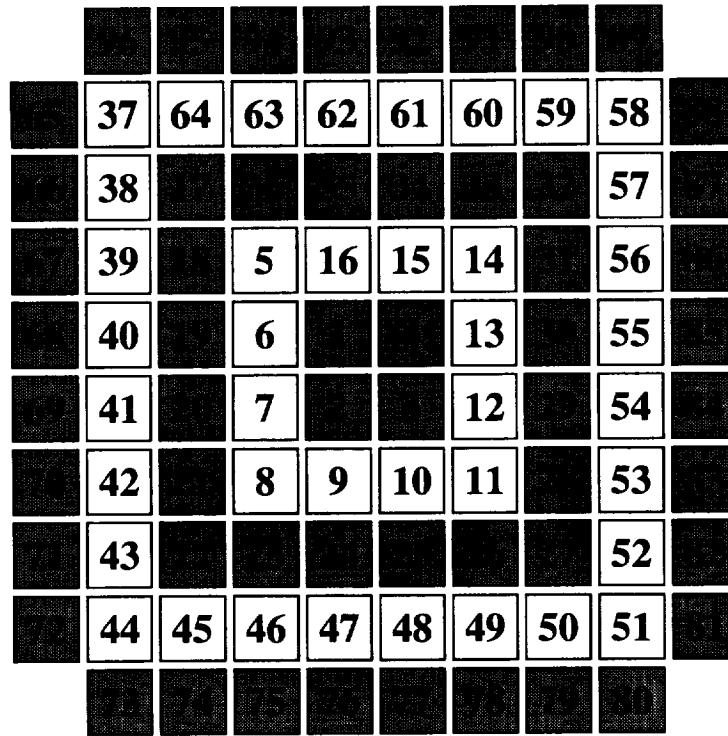


Figure 2.6: Heater arrangement of third-generation heater

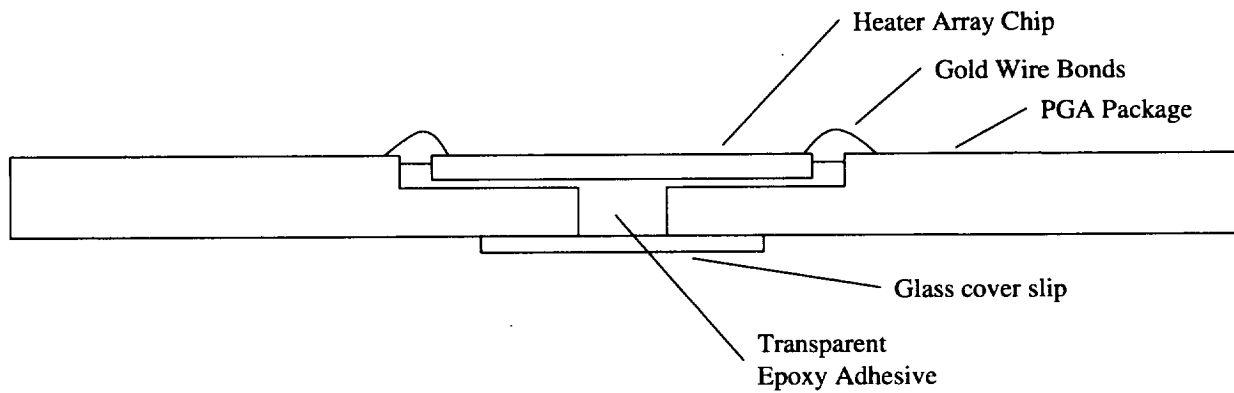


Figure 2.7: Schematic of quartz wafer mounted on the PGA package

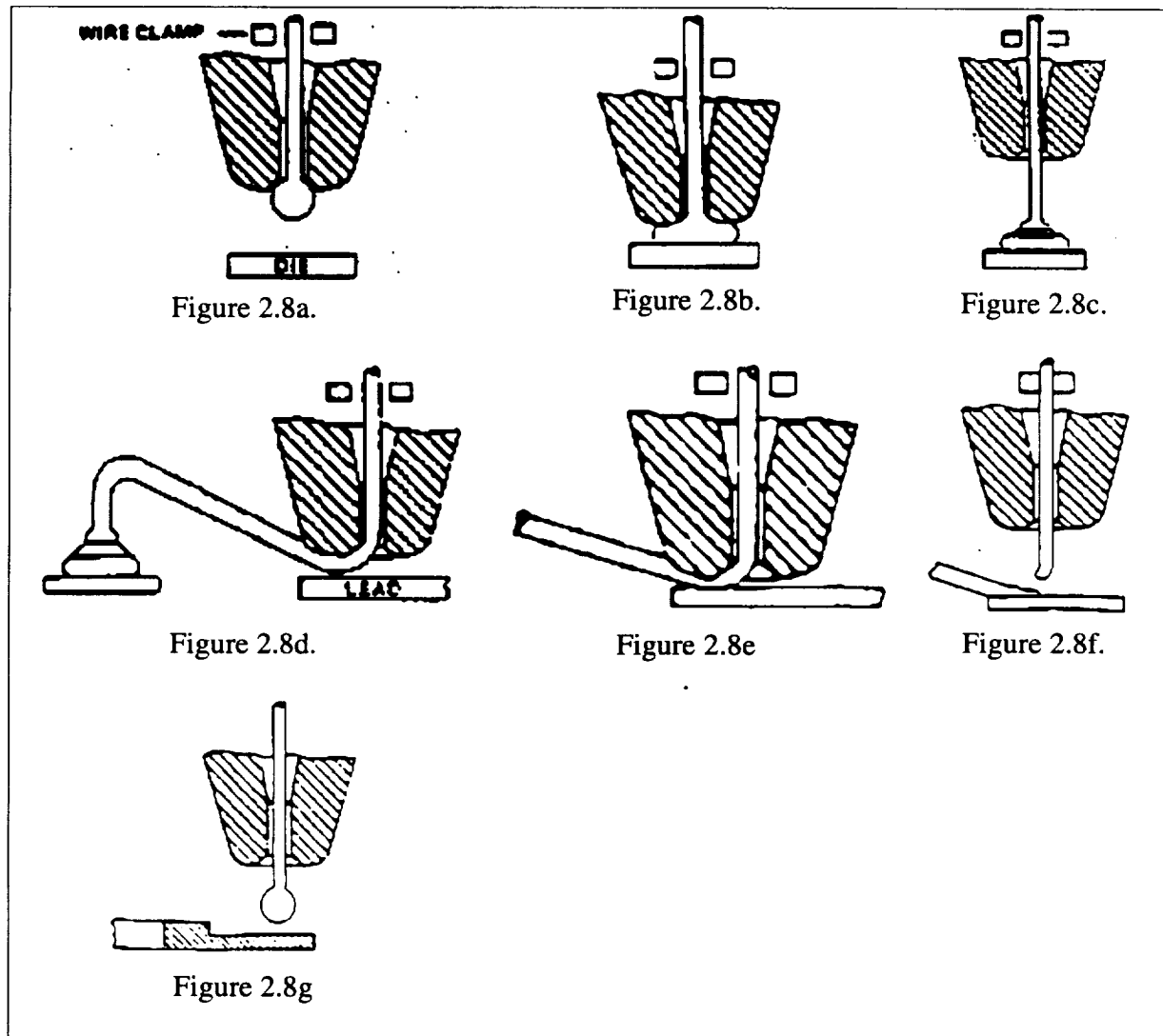


Figure 2.8: Wire bonding method

## **CHAPTER 3: CONSTANT TEMPERATURE CONTROL SYSTEM AND DATA ACQUISITION**

A microscale heater array has been developed that contains 96 resistive heater elements. Both the heat flux and the temperature from the surface can be measured from these elements if an appropriate control and data acquisition system is used. A feedback control system has been developed which maintains each heater element at a constant temperature. Two different data acquisition systems are used to digitize the heater voltages, so that the heat flux from the heater can be determined.

Figure 3.1 shows a block diagram of the elements of the control and data acquisition system. The temperature of each of the 96 platinum resistance heaters in the array is held constant by one of a group of 96 feedback control circuits. A microcontroller interfaced to a personal computer through a RS-232 serial port controls the temperature setting of each of the 96 feedback control circuits. Each heater/feedback control combination must be calibrated so that a given control signal from the microcontroller corresponds to a known heater temperature. One of two different analog-to-digital data acquisition systems is used to gather information on the heat transfer from each platinum heater. One is able to collect data at lower speeds, and transfers data to a PC through a parallel interface. The other is capable of higher speeds, and transfers data to a PC through a PCMCIA digital I/O adapter.

### **3.1. FEEDBACK BOARD**

Each heater in the array has a nominal resistance of  $1000\ \Omega$  and the resistance changes with temperature by about  $2\ \Omega/^{\circ}\text{C}$ . Thus, if the temperature of a heater remains constant, then the resistance of the heater will remain constant. The control circuits keeps the temperature

constant by sensing the resistance in a heater and adjusting the power dissipation of the heater so that the resistance remains constant.

Constant-temperature hot wire or hot film anemometers detect fluid velocity by detecting changes in the heat transfer coefficient on the surface of a thin wire or metal film. This is achieved by holding the wire or film at a constant temperature using a feedback controller. The device uses a Wheatstone bridge and a differential amplifier to accomplish this task. The feedback control for the constant temperature heaters works on the same principle. Following is a description of the control circuit operation.

Figure 3.2 shows a functional schematic of the electronic circuit that is used to maintain a heater at constant temperature. Its major elements are a Wheatstone bridge circuit, a feedback amplifier attached to the Wheatstone bridge, a control resistance circuit, and a buffer amplifier for the A/D output. The bridge is said to be balanced when  $V_1 = 0$ . This occurs when the ratio between  $R_4$  and  $R_2$  is the same as between  $R_3$  and  $R_1$ . Once the bridge is balanced, it will remain balanced regardless of how the supply voltage,  $V_s$ , changes.

The feedback circuit maintains a heater at constant temperature by detecting bridge imbalance and regulating the current through the bridge in order to bring it back into balance. The op-amp output will increase or decrease so that the heater will heat up or cool down until the heater reaches the resistance necessary for the bridge to balance. The op-amp output drives an NPN transistor which provides enough current gain to drive the heaters.

### **3.1.1. Voltage-Controlled Resistance**

The present microscale heater array design uses 96 feedback control circuits similar to the one shown in Figure 3.2 to maintain each of the heaters in the array at a constant temperature. In simplest conceivable feedback circuit design, the set point of the heater is changed by



adjusting a potentiometer on the high-impedance side of the bridge. This design would require adjusting over 96 potentiometers by hand each time a different temperature or a different heater was used in an experiment, and the size of the potentiometers would be significant. Calibrating such a system for a number of different temperatures, and for a number of different heaters, would be tedious and time-consuming. The authors sought an alternative method of adjusting the bridge resistance automatically.

One possible method of adjusting the bridge resistance is to use a voltage-controlled resistance, where the voltage is supplied by a computerized control system. The first method that was considered was a circuit using a JFET voltage-controlled resistance device produced by Temic. These devices are designed for applications where the voltage across the device is a low-level AC signal, but were found to operate linearly in the DC range. However, they displayed a large temperature dependence, which made it impossible to accurately calibrate the resistance of the device to the control voltage.

Figure 3.3 shows a schematic diagram of an analog multiplier configured as a voltage-controlled resistance. The operation of an analog multiplier is governed by the equation

$$\frac{(X_1 - X_2)(Y_1 + Y_2)}{10V} + Z = W$$

where  $X_1$ ,  $X_2$ ,  $Y_1$ ,  $Y_2$ , and  $Z$  are inputs, and  $W$  is the output.

The circuit illustrated in Figure 3.3 acts as a voltage-controlled resistance connected between P1 and GND. The equivalent resistance of the circuit,  $R_{eq}$ , can be determined through circuit analysis to be

$$R_{eq} = \frac{R_1}{1 + \frac{Y_1}{10V}}$$

Thus,  $R_{eq}$  varies from  $R_1$  when  $V_{cmd} = 0$  V, to  $\frac{R_1}{2}$  when  $V_{cmd} = 10$  V. A derivation of the equivalent resistance is given in Appendix A.

Figure 3.4 shows a schematic of a complete feedback control circuit for controlling a single heater element. The potentiometer R106 is used to adjust the offset voltage of the op-amp. The effect of offset voltage on the circuit is discussed in Section 3.7. . The pair of op-amps U3A and U3C invert and offset the  $V_{cmd}$  signal before it reaches the multiplier chip, so that when  $V_{cmd} = 0$ ,  $Y1 = +10$ V, and when  $V_{cmd} = +10$ V,  $Y1 = 0$ V. This arrangement is necessary because the heater resistance,  $R_h$ , and thus the heater temperature, will be maximum when  $Y1$  is zero Volts. Since the control unit which provides the  $V_{cmd}$  signal may default to zero volts in a power-off state, it was feared that if  $V_{cmd}$  was connected directly to  $Y1$ , a power-loss to the control unit might result in  $R_{eq}$  being forced to its maximum value. This condition would force the heater to its maximum temperature, and an expensive heater burn out might result.

The specifications of some important components of the control system are provided in Appendix E.

### **3.1.2. Transient Response**

The transient response rate of the circuit was tested by observing the circuit output when step-changes in the equivalent resistance were produced by applying a square-wave signal to the  $V_{cmd}$  input. A signal generator was used to produce a square wave with varying frequency and amplitude. An oscilloscope was used to measure and compare the square wave input and the circuit response. The circuit was attached to a platinum heater during the test, so the results represent the electrical response of the circuit combined with the thermal response of the heater in air. The frequency was increased until the feedback circuit was no longer able to respond with the necessary change in heat-flux.

Figure 3.5 shows the circuit response to a  $V_{cmd}$  square wave near the maximum transient response frequency. The figure shows the transient response time of the circuit and the heater to a step change in either the heat-transfer coefficient or to a step-change in heater temperature. This response time is one way to characterize the transient response capability of the circuit. This response time was not measured.

Another way to characterize the transient response is to increase the frequency of the square wave signal until the magnitude of the transient voltage response decreases to  $1/\sqrt{2}$  times the low-frequency value, or from about 7 divisions wide to about 5 divisions wide on the oscilloscope screen. The maximum transient response frequency according to this measurement was 16 kHz.

### **3.2. COMPUTER CONTROL BOARD**

A microprocessor control board was designed that is able to regulate the control voltage for each of up to 160 feedback circuits. The board is able to store in memory the control voltages that correspond to 16 different temperatures, and it is programmed by transmitting control strings to it from a personal computer via an RS-232 serial interface.

Each voltage table contains an array of control voltage values for up to 16 control cards with 16 channels per card. However, the present computer control board design is able to physically control only 10 feedback cards. Control voltages are selected over a zero to ten volt range, with 12 bit resolution. Control strings for various task are listed and explained in Table 3.2.

Function	Description	Command
Store	Store a 12-bit D/A value in one of the 16 voltage tables. Value is stored as the most significant 12 bits of a 16-bit value.	'S' + ADDR + VAL ADDR = 4 hex digits (See Table 3.3) VAL = 4 hex digits. 3 digits ranging from 000h to FFFh. 0 appended to end EXAMPLE: SC000FFF0
Table	Select 1 of 16 voltage tables.	'T' + TABLE NUMBER TABLE NUMBER = 1 hex digit, from 0h to Fh. EXAMPLE: T0
Calibrate	Selects 1 of 16 special pre-set voltage tables, for diagnostic purposes.	'C' + CAL TYPE CAL TYPE = 1 hex digit, from 0h to Fh EXAMPLE: C0
Read	Outputs the values currently stored in the selected voltage table. Values are output as 16-bit addresses and 16-bit values. The most significant 12 bits of the value represents the control voltage value.	'R' + TABLE NUMBER TABLE NUMBER = 1 hex digit, from 0h to Fh.  EXAMPLE: R0
Error	Outputs a list of the last 16 communication errors, and clears the error list.	'E' EXAMPLE: E
Voltage	Reads the 4 system voltage (Implemented in software, not in hardware)	'V' EXAMPLE: V
Reset	Resets the microcontroller program to address E000, the initialization routine of the machine code program. Equivalent to pushing the reset switch.	'X' EXAMPLE: X
Pulse	Sets the length of the voltage pulse that is sent to the feedback control cards.	'O' + VAL VAL = 4 hex digits. Default is 0080h EXAMPLE: P0080
Delay	Sets the delay in the program loop. Large delay updates the feedback control circuits more slowly.	'D' + VAL VAL = 4 hex digits. Default is 0004h EXPAMPLE: D0004

'h' on the end of numbers indicates that the number is a hex value

Table 3.2: Computer Control Board commands

				Table Select				Card Select				Channel Select				
Bit Value	1	1	0	ts	ts	ts	ts	cs	cs	cs	cs	ch	ch	ch	ch	0
Hex Digit	1				2				3				4			

Table 3.3: Computer Control Board addressing scheme

The computer control board programs each of the 96 feedback control circuits with the correct control voltage by scanning through all 96 control circuits and transmitting voltage pulses which are stored in a capacitor on each feedback circuit. Rather than having 96 separate signal connections from the computer control board, a multiplexing scheme is used, where a single wire carries a train of voltage pulses to all the boards and an address bus directs the voltage signals to the correct feedback circuit.

Figure 3.6 shows how the multiplexing scheme works. A single wire carries a train of voltage pulses to all 6 feedback cards. A card-select bus that goes to each feedback card goes high whenever a voltage pulse is to be sent to a feedback circuit on that card. A channel-select bus goes to all the cards and directs the voltage pulse to one of the 16 feedback circuits on the selected feedback card. This is accomplished using a 16-channel multiplexer on each feedback card.

When the computer control board is first turned on, the microprocessor initializes all the voltage tables to zero volts. Otherwise, the memory that holds the voltage tables would assume random values. Some of these values may be high, which would cause these heater elements to be driven to a very high temperature. This could cause a heater array to be ruined if a heater burned out.

The voltage pulses are stored by a capacitor that is associated with each feedback circuit. When a feedback circuit is not selected to receive a voltage pulse, the output impedance of the multiplexer is very high, so that the capacitor voltage that has been set by the voltage pulse does not leak down.

Figure 3.7 shows the order that the channel-select and card-select addresses are incremented as the computer control board scans through the feedback circuits. The  $V_{cmd}$  signal

is adjusted at each step to match the control voltage that is stored in memory for a given feedback circuit. A PULSE signal determines how long the card select pulses last, which in turn determines how long the multiplexer applies the control voltage to the selected feedback circuit. The channel-select value that is sent on the 4-bit wide channel-select bus is shown at the bottom of the figure.

The microprocessor on the computer control board performs both the task of transmitting the voltage pulse train, and also of downloading control strings from the RS-232 port. When the RS-232 data is being read, the control board temporarily stops transmitting the pulse train. Instead, the last voltage level in the train is transmitted to all the feedback circuits while the microprocessor is downloading commands. For a given temperature, the necessary control voltages can vary significantly from heater to heater. Therefore, if a new temperature table is being uploaded while an old one is set, it can cause the control voltage to get stuck on a fairly high value and cause many of the heater to jump to a very high temperature. This tends to initiate boiling on the surface at many points, which

A better method of switching from temperature to temperature is to leave one voltage table at zero volts and download up to 15 other temperatures to the other voltage tables. Then, the "change table" command can be used to change temperature very quickly.

A watchdog circuit on the computer control board will cut off power to the heater array if the microprocessor stops functioning. All power to the heater array is routed first to the computer board, then through a relay, and finally into the back plane where it is distributed to the computer control circuits. An output from the microprocessor is required for the relay to be energized and power to be supplied to the heater array. A complete schematic of the computer control board is given in Appendix F.

### 3.3. A/D SYSTEMS

Two systems are available for acquiring heat transfer data. Both systems digitize the voltage across the heaters and transfer the digitized data to a personal computer. The first system is a commercial system which transfers data through the parallel port of the PC, and the second system, which is integrated into the control system, transfers data through a PCMCIA port into a laptop computer.

#### 3.3.1. IOtech A/D system

The first system, which is sold by IOtech, consists of an analog to digital unit, a DaqBook/216, an expansion chassis, the DBK41, and up to 10 DBK12 multiplexer cards.

The main board can digitize up to sixteen single-ended voltage inputs with 16-bit resolution. Gains of 1, 2, 4, or 8 can be selected for each input channel. With a gain of one, the unit can digitize voltage levels between 0V and 10V. A gain of 1 is used for data acquisition, because the heater voltage will usually be between 0V and 10V. Voltages are digitized and transferred to a PC through the parallel interface at a rate as high as 100,000 samples/second. Up to 16 channels can be sampled on the main unit by multiplexing the A/D converter.

The DBK41 expansion unit provides space for 10 IOtech expansion cards, and connects to the input port of the DaqBook/216. As many as 10 DBK12 multiplexer cards can be installed in the DBK41 unit. Each DBK12 multiplexes up to 16 single-pole voltage inputs into a single input channel of the DaqBook/216. Thus by using 6 DBK12 multiplexer cards, 96 channels of data can be scanned using a single DaqBook base unit. Figure 3.8 shows how the components of the DaqBook system are connected.

Since the analog to digital conversion is accomplished by scanning the multiplexed channels at a rate of 10  $\mu$ s/channel without a simultaneous sample-and-hold on the inputs, each

sample within a scan is taken  $10\mu\text{s}$  before or after the adjacent samples. This time skew must be considered when interpreting the results from this method. This time skew is illustrated in Table 3.4. Each column represents a 1 ms time interval, and an X marks the periods in time where each channel is digitized.

Appendix C gives specific details about the hardware and software configuration of the IOtech data acquisition system.

Channel Number	Time Interval (ms)				
	10	20	30	....	960
1	X			....	
2		X		....	
3			X	....	
....	....	....	....	....	
96					X

Table 3.4: Timing of DaqBook digitization

### **3.3.2. High-speed A/D board**

A second A/D system was designed at the University of Denver specifically for the constant-temperature control system. It is able to digitize 96 heater voltages at a rate of up to 10,000 samples/s per channel, yielding 10,000 heat flux maps per second. It has the potential for up to 160 input channels using a total of 5 A/D cards. Three cards are used with the current system, providing a total of 96 input channels. One card acts as a master card to govern the timing of itself and the other cards in the system. Each card services two feedback controller cards, and digitizes the voltages from 32 channels with 12-bit resolution. Each board contains six  $256\text{k}\times 4$  static ram chips, which provide memory storage for 524,288 12-bit values, or 16,384 values per channel. The A/D board can sample data at rates of 1.25, 2.5, 5, or 10 ksamples/s per channel. Thus, the board can store 1.64 seconds of information using the maximum sampling rate, or 13.1 seconds at the minimum sampling rate.



A PCMCIA digital I/O card is used to connect a laptop computer to the A/D system for downloading data from the buffer on each A/D card into personal computer memory, where it can be manipulated and analyzed. The digital I/O card provides 13 digital inputs and 8 digital outputs for interfacing with the A/D system. Appendix C describes the digital I/O interface in detail, and discusses the procedure for downloading data from the A/D cards.

The data obtained by the high-speed A/D board is skewed in time. For instance, if the voltages are being sampled at 10,000 samples per second, all 96 heater voltages will be digitized every 100  $\mu$ s. The three A/D cards in the system each contain 2 separate A/D converters, for a total of 6 A/D converters. Each of the A/D converters digitizes 16 channels in that 100  $\mu$ s period. The digitization is done without a simultaneous sample-and-hold circuit, so these 16 channels are skewed in time. The timing of the data acquisition process is illustrated in Table 3.5 for a 100 kSample/s case. In this table, each group represents the 16 heaters that are sampled by a single A/D converter.

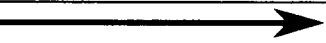





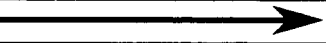
1 Time Increment = 6.26 $\mu$ s						
Total of Time increments = 100 $\mu$ s						
Group #	Time Increments ( $\mu$ s)					
	6.25	12.5	18.75		93.75	100
0	Ch. 1					
		Ch. 2				
			Ch. 3			
						
					Ch. 15	
						Ch. 16
1	Ch. 1					
		Ch. 2				
			Ch. 3			
					Ch. 15	
						Ch. 16
						
9	Ch. 1					
		Ch. 2				
			Ch. 3			
						
					Ch. 15	
						Ch. 16

Table 3.5: Timing of high-speed A/D cards

### 3.4. PHYSICAL ARRANGEMENT OF CONTROL HARDWARE

#### 3.4.1. Backplane

Most of the control and data-acquisition components are housed in a card-cage chassis that has slots for up to 21 printed circuit boards to slide into a motherboard. Only 16 of the slots are used by the motherboard, which has card-edge connectors for 10 feedback controller cards, 5 A/D cards, and 1 computer control card. The system was designed to control up to 160 heaters,

but at present, only 96 heaters are being controlled, so only 6 feedback cards, 3 A/D cards and 1 computer control card are installed. The cards slide into the card cage and connect to card edge connectors on a printed circuit board backplane. The backplane also has connectors for an RS-232 port, a PCMCIA port, an alternative A/D device such as the DaqBook, and the heater array itself. Figure 3.9 shows the arrangement of card-edge connectors on the backplane board.

One set of ribbon cables connects each feedback card in the card cage to 16 heaters. Another set of ribbon cables connects the A/D inputs to the heater voltage outputs on each feedback card. Each ribbon cable connects 16 A/D channels. Figure 3.10 shows the arrangement of power, A/D, computer control and heater connectors on the back of the board.

#### **3.4.2. Decoding board**

It may be useful at times to collect data from or supply power to only a specific group of heaters on the heater array. This may be useful when testing equipment, or when studying the effects of the size of the heated area.

For instance, if the 96 heaters in the array are mapped to the feedback circuits in an arbitrary fashion, then it might be necessary to use all 6 feedback control cards and 3 A/D cards to sample only the center 16 heaters in the array. It is much easier to work with just one or two feedback cards or A/D cards when troubleshooting the system. It is also more intuitive to troubleshoot when the first feedback card controls the center 16 heaters, the next one controls the next ring of 16 heaters, etc. Ease of troubleshooting is a significant issue with a new system such as this. For these reasons, it is desirable to map a specific group of heater to a specific feedback card.

To accommodate this possibility, a decoding card is used to physically map specific heaters to specific feedback circuits. The backplane is connected to the decoding card, and the

decoding card is connected to the heater array. Connections are made on the decoding card between the heater connections and the backplane connections using wire-wrap terminals. Wire-wrap connections allow any heater on the array to be mapped to any control circuit using a simple wire-wrap operation.

A circuit board like the one shown in Figure 3.11 is used to connect the four 50-conductor ribbon cables from the heater array to 6 26-conductor ribbon cables that connect to the backplane. The ground leads from the heater array are connected to the aluminum bus-bar to provide a uniform ground voltage, and the power leads are connected to the 26-pin ribbon-cable connectors using wire-wrap connections. The ground null circuit shown in the figure is used to maintain a constant ground voltage relative to the feedback cards. The need for this circuit is explained in Section 3.5.

The printed circuit board shown in Figure 3.12 is used to connect the 4 50-conductor ribbon cables to the heater array. The ribbon-cables terminate in card-edge connectors which snap onto the four sides of the circuit board. The card edges are connected to a PGA socket in the center of the circuit board. The PGA package which holds the heater array snaps into this PGA socket. The circuit board and heater array are mounted on the test chamber using an aluminum backing plate with an insulating rubber pad between the backing plate and the circuit board. 4 machine screws pass through the 4 holes in the circuit board and hold the backing plate, rubber pad, and circuit board to the experiment apparatus.

### **3.5. GROUND NULL CIRCUITRY**

The method of grounding the individual heaters in the heater array can be very important. This issue has been mentioned briefly in the chapter on heater construction. In this section, it is discussed in more detail, potential solutions are presented, and the final solution is discussed. To

resolve this issue, the electronic circuit design, the printed circuit board layout, and the heater array design all must be considered.

Figure 3.13 shows a schematic of a grounding scheme for the heater array. The ground shown is the ground that is common to the feedback control circuit. The multiple parallel impedances  $R_1$  through  $R_n$  are the  $n$  heater elements, and  $R_g$  is the impedance in series with all of the heaters.  $R_g$  is generally so small that if only a single heater is operating, such as when calibrating one heater at a time,  $R_g$  is negligible. However, when multiple heaters are operating, the current from all the heaters flowing through  $R_g$  results in a potential across  $R_g$ ,  $V_g$ , that adds to the voltage across each of the heaters in the array.

Figure 3.14 shows how the voltage across the ground resistance affects the feedback loop circuit. Since this voltage is much smaller than the voltage at the top of the bridge, which is always by design at least 200 mV, it has an insignificant effect on the current flowing through the heater side of the bridge. The only other effect is that it raises the voltage on the inverting side of the op-amp by the amount  $V_g$ . Therefore, since the effect on current is negligible, the voltage can be applied in an analysis as shown in Figure 3.15, where its only effect is to offset the inverting input, as shown in Figure 3.16. The result is that the circuit responds as if the op-amp offset voltage was more positive by the amount  $V_g$ . This fact can be used with the offset voltage analysis to determine the effect of  $V_g$  on the temperature uncertainty.

In fact, positive offset tends to de-stabilize the circuit, so that a feedback control circuit may not turn on at all if  $V_g$  is too large. This is because the higher voltage that is detected across the heater causes the feedback circuit to respond as if the resistance was much higher. Since there is only 100 mV across the heater initially, a 2 mV offset can cause the heater to appear to have a resistance  $20\Omega$ , or  $10^\circ\text{C}$ , higher than it really has. That means it will not turn on until the

temperature of the heater gets at least 10 degrees below the setpoint, which will not happen if the surrounding heaters are actively keeping the temperature at the setpoint temperature. The effects of the ground resistance was discovered when certain heaters refused to turn on when multiple heaters were operating, but turned on linearly when that heater was operating by itself.

Sources of the ground impedance include the doped silicon substrate, the contact between the silicon and the silver epoxy, the contact between the silver epoxy and the ground lead, and the ground lead itself. These impedances were measured, and the ground wire was shown to be the most significant resistance.

Three solutions to this problem were considered. The first option was to assemble the heater array and control system so that the ground lead is a very heavy gauge wire connecting to the system ground. The second option was to attach the ground lead to an amplifier that would sense the ground potential on the substrate and drive that potential to zero by changing the current through the ground lead.

These first two solutions eliminate the problem with ground lead resistance, but they do not deal with the resistivity of the doped silicon substrate. A simple experiment showed that this resistivity still causes an unacceptable offset in the ground potential.

Figure 3.17 shows a schematic of the experiment that measured this ground potential variation. A 5V power source was used to power 16 heaters located in one quadrant of the array. The ground potential of several other heaters at various locations around the array was measured by connecting a multimeter to the heater power leads. Since the multimeter input impedance is much larger than the heater impedance, this is an accurate method of measurement. The results showed that the heater ground potential varied by as much as 1 mV across the heater array. The case where only 16 heaters are running at half the maximum operating voltage is conservative,

and even 1 mV of offset voltage is unacceptable, according to the analysis in Appendix A, so a heater array design that uses the substrate as a ground is not practical.

A revised design using a quartz substrate and an aluminum ground plane on the top layer was analyzed to determine if a similar ground potential variation would be manifest. Knowing the dimensions of the aluminum ground plane and the amount of current that would be expected in this ground plane, it was determined that this design would in fact exhibit the same ground potential problem as the doped silicon substrate.

The final solution was to redesign the heater array and the control system so that each heater has its own ground lead which would connect to the feedback card, so that there would be no ground offset voltage due to current from other heaters. The final solution involved providing an individual ground for each heater which leads to a thick aluminum block. A simple amplifier circuit, like the one shown in Figure 3.18, is used to drive this ground to the same potential as the ground of the motherboard which connects the feedback boards.

### **3.6. CALIBRATION METHOD**

Calibration is accomplished by a series of steps that are performed while the heater array is placed in a constant-temperature liquid bath. First the bath is allowed to reach steady state at a set temperature. Next, the  $V_{cmd}$  level applied to one of the feedback circuits is gradually increased, causing  $R_{eq}$  to increase, until the bridge starts to balance and the heater power begins to increase. The power into the heater is measured, and  $V_{cmd}$  is increased until the heater voltage reaches a certain low level. The value of  $V_{cmd}$  at which this minimum voltage level is detected is saved in a file on the PC, so that it can be used in subsequent experiments. The  $V_{cmd}$  value for that heater is reset to zero, and the preceding steps are repeated at the same temperature for every other heater in the heater array.

It is desired to minimize the heater temperature uncertainty during this calibration procedure. When the circuit applies power to the heater, the heater temperature will rise slightly higher than the bath temperature. The magnitude of this temperature rise depends on the thermal conductivity of the substrate and the heat transfer coefficient seen by the heater surface. This temperature rise introduces an error into the calibration, because the heater temperature will not match the bath temperature. Temperature rise vs. heater power can be measured experimentally for low power levels by applying a known current to the heater and measuring the heater resistance at various small current levels. Knowing the maximum acceptable error due to heater temperature rise, a maximum calibration power level can be selected from this experimental data. At this power level, the feedback control will be operating in its linear range, but the heater temperature will be negligibly higher than the constant temperature bath. The heater voltage that results in this maximum calibration power level will be referred to as a threshold voltage. All the heaters will be calibrated at this threshold voltage level, to provide the minimum uncertainty due to both heater temperature rise and control circuit non-linearity.

The method of calibration is discussed in more detail in the Section 4.1, Calibration Apparatus

### **3.7. TRIMMING**

Several adjustments must be made to each individual feedback circuit to ensure accurate temperature control. The purpose of these adjustments is to ensure that the voltage-controlled resistance does not change when the current flowing through the bridge is very small. When the threshold voltage is being applied to the heater during calibration, the voltage-controlled resistance must be very close to its value at the maximum voltage, and all values in between the threshold voltage and maximum voltage. Unfortunately, it is difficult to measure the magnitude



of the voltage-controlled resistance during operating conditions. A more effective approach seems to be to observe the response of the circuit to changes in the control voltage when a constant resistance is connected in place of the heater.

Ideally, the operational amplifiers used in the feedback control circuits would have zero offset voltage and infinite gain, and the control resistance would not change as the heater power changes. Therefore, the response of the feedback circuit with ideal components would be similar to that shown in Figure 3.19. The control voltage where the sharp jump in heater voltage occurs is the point where the bridge is balanced. If the operational amplifier has a finite, negative offset voltage, the circuit response would appear as shown in Figure 3.20. The finite offset voltage causes the circuit to turn on more gradually, even when the heater resistance is constant. This is not desirable when a heater is calibrated, because of the temperature error that is evident in Figure 3.20.

Figure 3.20 shows two constant-resistance lines on a plot of heater voltage vs. control voltage. For each of the resistance values, the control circuit will calibrate to a control voltage of  $V_1$  when the resistance used is  $R_1$ , and of  $V_2$  when the resistance used is  $R_2$ . It can be seen, however, that the resistance that the heater actually operates at is determined by the heat flux. For instance, at control voltage  $V_2$ , the heater will maintain resistance  $R_2$  when the heater voltage is low, but when the heater voltage reaches its maximum limit, the heater resistance will be  $R_1$ . Therefore, the calibrated at  $V_2$  might vary in resistance from  $R_1$  to  $R_2$ . This introduces an uncertainty into the temperature calibration. The difference between  $R_1$  and  $R_2$  depends on the width of the constant-resistance line and the relationship between changes in control voltage and heater resistance.

The purpose of trimming the adjustments of each feedback circuit is to minimize the width of the constant resistance curve between  $V_{\text{thresh}}$  and the maximum heater voltage, so that the uncertainty in the temperature calibration is minimized. Trimming is accomplished by connecting constant value resistors place of the temperature-sensitive heaters, and varying  $V_{\text{cmd}}$  to obtain a real-time plot of a constant resistance curve like the ones in Figure 3.20. Each circuit is then adjusted to minimize the error due to the width of the portion of the curve between the threshold voltage and the maximum voltage.

$V_{\text{cmd}}$  was varied by a signal generator which supplied a triangle wave to the  $V_{\text{cmd}}$  input on each feedback circuit. The amplitude of the signal generator is set to the value of the  $V_{\text{cmd}}$  change that would result in the maximum allowable temperature uncertainty. This signal was connected by unplugging the multiplexer which distributes the control voltage pulses from the computer control board to each feedback circuit on a card. A socket connect was then used to distribute the external  $V_{\text{cmd}}$  signal to the multiplexer outputs. The multiplexer outputs lead to the inputs to the analog multiplier that regulates the voltage-controlled resistance. An oscilloscope is connected to both the  $V_{\text{cmd}}$  signal and to the heater voltage. The oscilloscope is set for X-Y mode, so that  $V_{\text{cmd}}$  controls the X-axis and the heater voltage controls the Y-axis. 1/8 watt metal film resistors are used in place of the heater, because the resistance of metal film resistors is much more stable than the resistance of carbon resistors.

There are two adjustments which affect the performance of the feedback circuit at low voltage levels: op-amp offset and analog multiplier output offset. There is actually an analog multiplier input offset voltage as well, but it is not independent from the other two adjustments, so that the effects of the multiplier input offset can be eliminated by adjusting the other two parameters.

The multiplier output offset could be most easily eliminated by connecting the circuit shown in Figure 3.21 to the Z input on the analog multiplier. The Z input is added onto the W output, so that any offsets in W could be eliminated by an opposite offset applied to Z. This method provides the most general, most linear correction for the effects of the multiplier offset voltage.

However, a different method was chosen to compensate for this offset because of the history of the circuit design. When the circuit was first being tested, the effect of multiplier offset was not understood, and the circuit was modified through trial-and-error changes that improved the circuit performance. One change that improved the accuracy of the voltage-controlled resistance at low voltages was to supply a current level into the multiplier input node that changed with the level of  $V_{cmd}$ . At zero  $V_{cmd}$  it would be some positive value, and at maximum  $V_{cmd}$ , it would be zero. The circuit shown in Figure 3.22 provides a method of adjusting the magnitude of this current level. A large resistor is connected to a potentiometer which connects to the  $V_{cmd}$  input on the multiplier chip.

Unfortunately, the current level sometimes must increase with increasing  $V_{cmd}$ , if the multiplier output offset has the opposite sign. This was not considered until after the feedback cards had been manufactured, so no provision for this was made in the board. Thus, when a specific circuit requires this change, a connection on the board must be cut, and a new connection made to a different point on the circuit. Figure 3.23 shows the changes that must be made to the circuit. After this change is made, the bias current is positive when  $V_{cmd}$  is positive, and zero when  $V_{cmd}$  is zero. This bothersome step would not have been necessary if the trimming method shown in Figure 3.21 had been used instead to compensate for the multiplier offset. That method will adjust for either positive or negative input offset voltage.

The following steps are followed the first time the circuits are trimmed. During these steps, the constant-resistance curves for the given resistors is plotted using a power supply, signal generator and oscilloscope.

1. A high-value resistor is connected in place of the heater element
2. The op-amp offset voltage is adjusted until the constant-resistance curve exhibits minimum uncertainty. The  $V_{cmd}$  level is high, because the heater resistance is high, so the multiplier output offset adjustment will have no effect. This insures that when the multiplier output offset is adjusted, it will not affect the op-amp offset adjustment.
3. A low-value resistor is connected in place of the heater element.
4. The multiplier offset is adjusted until the constant-resistance curve exhibits minimum uncertainty.
5. If this adjustment cannot be made, completed, it means that the multiplier output offset has the opposite sign, and the modification shown in Figure 3.23 must be applied to the circuit, and the circuit must be re-trimmed.

The adjustment in step 5 requires a different set of trimming steps to be followed for that circuit. The following steps must be followed.

1. A LOW-value resistor is connected in place of the heater element.
2. The op-amp offset voltage is adjusted until the constant-resistance curve exhibits minimum uncertainty. The  $V_{cmd}$  level is high, because the heater resistance is high, so the multiplier output offset adjustment will have no effect. This insures that when the multiplier output offset is adjusted, it will not affect the op-amp offset adjustment.
3. The op-amp offset voltage is adjusted until the constant-resistance curve exhibits minimum uncertainty. The  $V_{cmd}$  level is LOW, because the heater resistance is LOW, so

the multiplier output offset adjustment will have no effect. This insures that when the multiplier output offset is adjusted, it will not affect the op-amp offset adjustment.

4. A HIGH-value resistor is connected in place of the heater element.
5. The multiplier offset is adjusted until the constant-resistance curve exhibits minimum uncertainty.

Once the circuits are modified as needed, note should be taken of which circuits have been modified for a different multiplier offset polarity. When the circuits are trimmed again, this information can be used to determine which of the above procedures to follow when trimming a circuit.

### **3.7.1. Ground potential variations across the motherboard**

It was mentioned above that positive offsets in the circuit cause instabilities. These instabilities cause the control system to malfunction drastically. Even if the above trimming procedure is followed, variations in the ground potential across the motherboard can cause the circuits to behave erratically in spite of careful trimming.

Figure 3.24 shows the variations in ground potential across the motherboard after the control system is first turned on and allowed to warm up. The solid outlines depict the slots where feedback cards are plugged in. These variations are measured relative to the aluminum ground bar on the decoder board which serves as a common ground for the heaters in the heater array. The variations are due to large DC ground currents in the motherboard. The ground plane is not thick enough to prevent significant ground-plane voltage gradients. Feedback cards are typically trimmed in the last slot on the right, to allow easy access to the trimming adjustments. These boards are then operated in one of the six left-hand slots. Thus, an offset arises which cannot be accounted for.

This offset can be accounted for by adjusting the ground potential at the slot where the board is being trimmed to match the potential that it will see in the slot where it is operated. This can be done using the ground potential adjustment circuit on the decoder board. The potential of the aluminum ground bar is kept constant by the simple amplifier circuit in Figure 3.18. The offset between the ground bar and the motherboard is controlled by adjusting a potentiometer, R1. Thus, if a board will typically see a 1.5 mV ground potential relative to the aluminum ground bar, then the potentiometer is adjusted until the slot where the feedback board is being trimmed, typically the last slot on the right, measures 1.5 mV relative to the aluminum ground bar.

An additional adjustment must be made to insure that positive offset is avoided. When the system is operated, the right side of the motherboard is maintained at 2.0 mV relative to the ground bar. This insures a negative offset in the circuit so that it will be stable in operation. The temperature error due to this offset is discussed in Appendix A.

### **3.8. SOFTWARE**

A set of computer programs are used to send commands from the laptop computer to the computer control board and the A/D systems, and to receive and process A/D information. These programs allow the user to automate calibration, data acquisition and data reduction tasks. They were developed in the Microsoft Visual Basic 4.0 environment under Windows 3.11 running on a laptop PC. Three programs are used to accomplish these tasks. The important features of these programs are discussed here. Appendix B provides complete instructions for using these programs. A source code listing is provided in Appendix G. The source code can also be obtained from the author on floppy disk.

The programs were developed in the Microsoft Visual Basic 4.0 environment, using the 16-bit development tools. 32-bit programs might run faster when performing numerical calculations and when multitasking with other programs. However, 16-bit programs were developed so that they could run under earlier Windows operating systems, and because multitasking performance was not important for this application.

The "CAL" program is used to automate heater calibration. The "CONTROL" program automates data acquisition and data reduction. The "GRAPH" program provides a quick means of viewing heat flux maps immediately after data is acquired.

The "CAL" program provides a means of automatically calibrating all 96 heaters at any given temperature. A constant temperature bath maintains the heaters at a given temperature, and the "CAL" program uses a bisection algorithm to determine the command voltage that results in the heater output closest to the threshold voltage. After all 96 heaters are calibrated at a given temperature, the voltages are saved in a calibration file so that another program can use them to set the heater to the desired temperature.

The calibration algorithm is very similar to the numerical bisection method. The computer control board is able to output control voltages between zero and ten volts. For the first heater, the computer tests the heater output voltages at 0V, 5V, and 10V. If the heater voltage is not found to cross from below to above the threshold voltage in that range, then an error occurs and the program advances to the next heater. Otherwise, the program continues to subdivide the interval in which the heater voltage crosses the threshold, until it reaches some minimum subdivision. At that point, the average of the command voltages for the last two subdivisions is saved as a calibration point. The calibration file consists of an array of these points.

The "CAL" program does not use the custom A/D system for gathering data. The custom A/D system is design to sample all the channels simultaneously, and download data for the entire heater array over a period no less than 1.6 seconds. 1.6 seconds is much too long to wait between successive steps in the calibration routine. The DAQBook A/D system, however, allows data to be collected from a single heater for a very short period of time. A typical time between calibration steps is 1/5 sec.

The "CAL" program is able to detect certain errors in the calibration process and notify the user in an error message box. For instance, if a heater is shorted out or broken, and thus the resistance of the heater is near zero or infinity, the program will notify the user of an error and set the command voltage for that heater to zero. If the resistance of a heater is outside the controllable range of the feedback circuits, an error will also result.

The "CONTROL" program allows the user to specify a group of calibration files to use in setting the heater temperature. Once these files are selected, the temperature can be changed by either specifying one of the calibration files to set the heater to, or by specifying a temperature and having the program interpolate between the available calibration files to provide the correct output voltage for that temperature.

The program provides a means of automating the acquisition of data at many data points. A list of temperatures and data file names can be entered, and a time to wait between data points is specified. Thus, when it is necessary to gather 50 data points waiting 10 minutes between data points, this long and tedious process can be easily automated.

Either the DAQBook or the custom A/D system can be used with the "CONTROL" program to acquire data. The DAQBook system is used when data is taken at a lower rate over a long period of time, such as when acquiring average heat transfer data. The custom A/D system



is used when acquiring data at fast sampling rates higher than the DAQBook maximum rate of 1000 samples/s per heater.

Data is saved using two files with the same name but different extensions. The raw binary data files have a ".BIN" extension. These files contain only a stream of binary values without any information to allow the binary values to be interpreted. In order to interpret the binary file, a file with the ".TAG" file is used to save information necessary to interpret the binary file, such as how many data points are in the file, how many heaters were sampled, what order the heaters were sampled in, what sampling rate was used, and what temperature the data was sampled at. Other comments can also be saved in this file when the data is acquired.

The "CONTROL" program also provides a post-processing utility which is used to perform various data-reduction operations on the binary file. These operations are discussed in detail in Appendix B. These operations include various time and space averaging operations. This utility also provides some primitive means of automating a series of data reduction operations, so that an entire group of files can be reduced automatically.

The "GRAPH" program provides the capability of immediately viewing heat-flux data graphically, without having to go through any data reduction steps. This is useful when the system is being set up to insure that it is functioning properly before starting a long data-acquisition run. The program allows the user to select a ".BIN" file, and then converts the voltage data to heater power information. An animated map of the power being dissipated by each heater is then displayed in the program window, using a false-color technique to display the heater power as a color level.

## 3.9. FIGURES

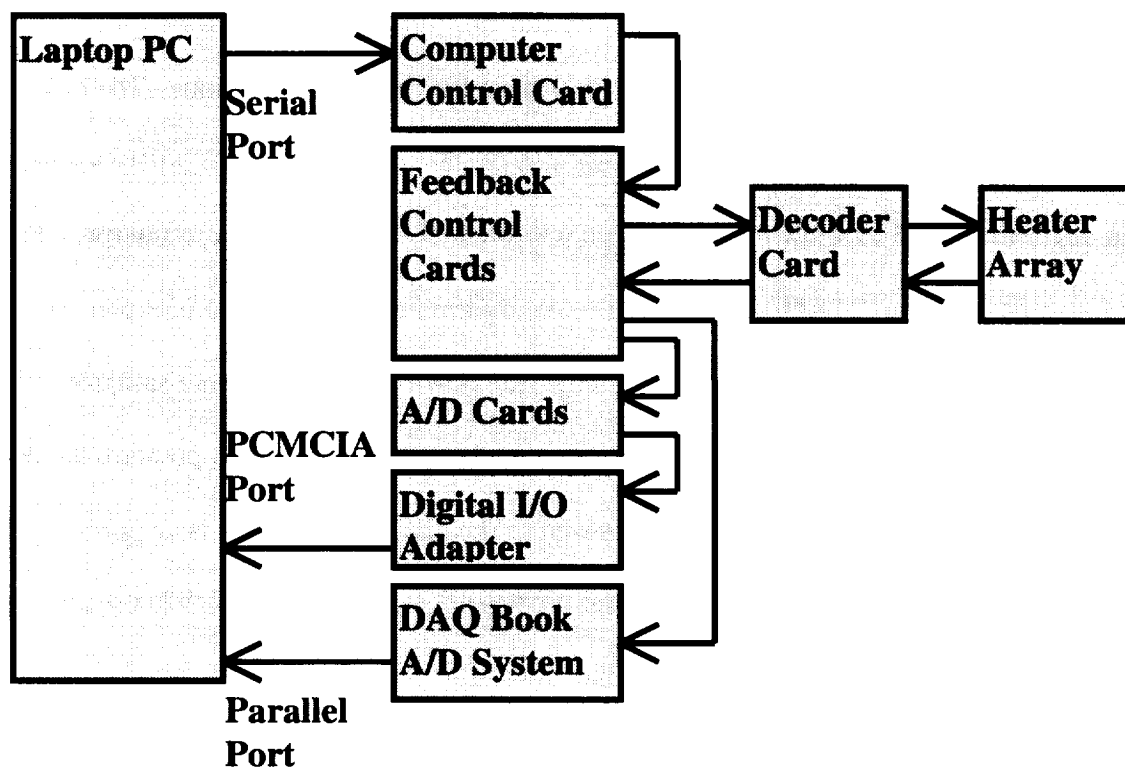


Figure 3.1: Schematic of control and data acquisition system components

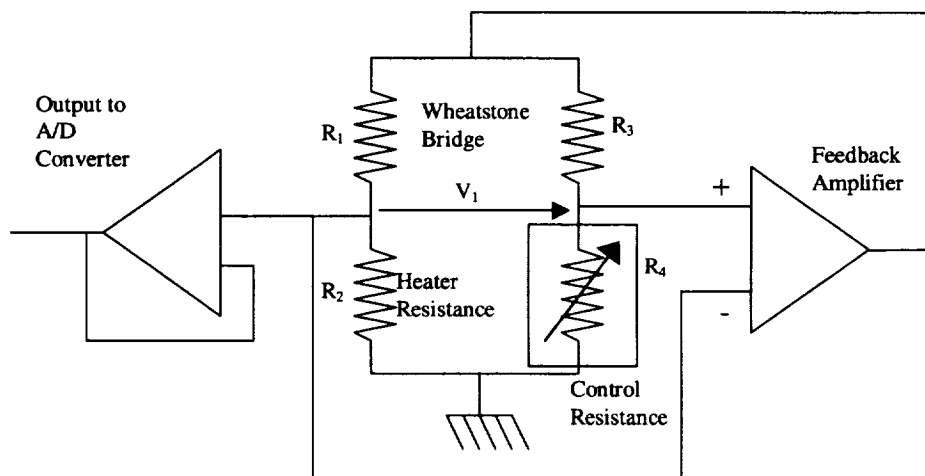


Figure 3.2: Functional schematic of feedback control circuit.

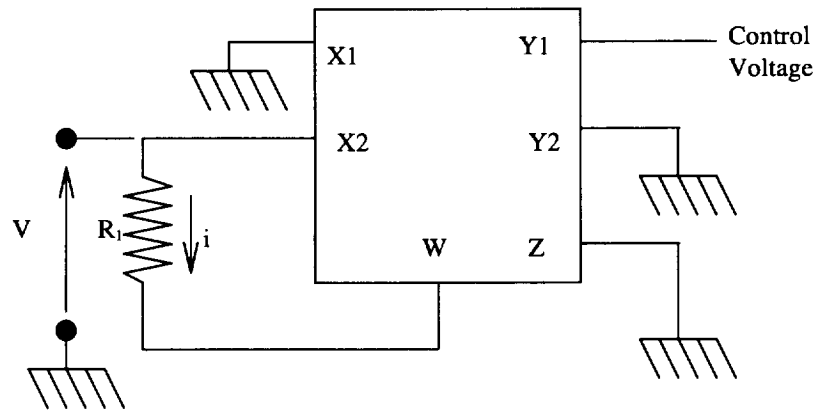


Figure 3.3: Analog multiplier configured as a voltage-controlled resistance

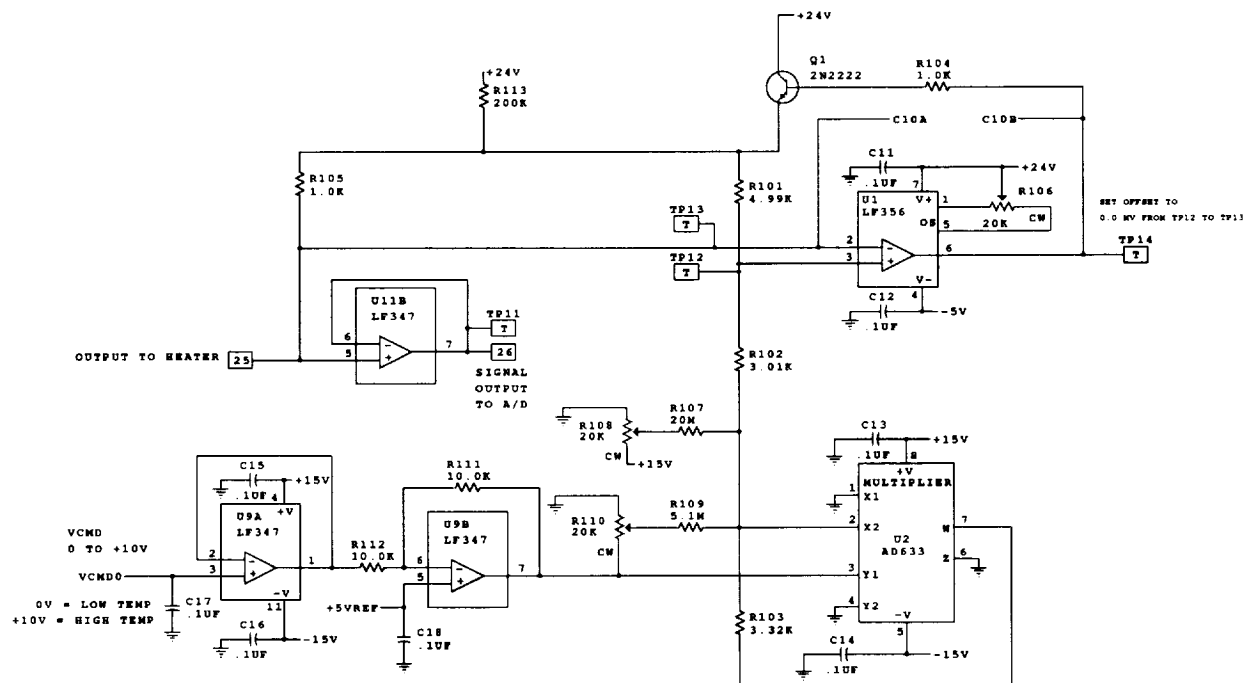


Figure 3.4: Complete schematic of feedback control circuit.

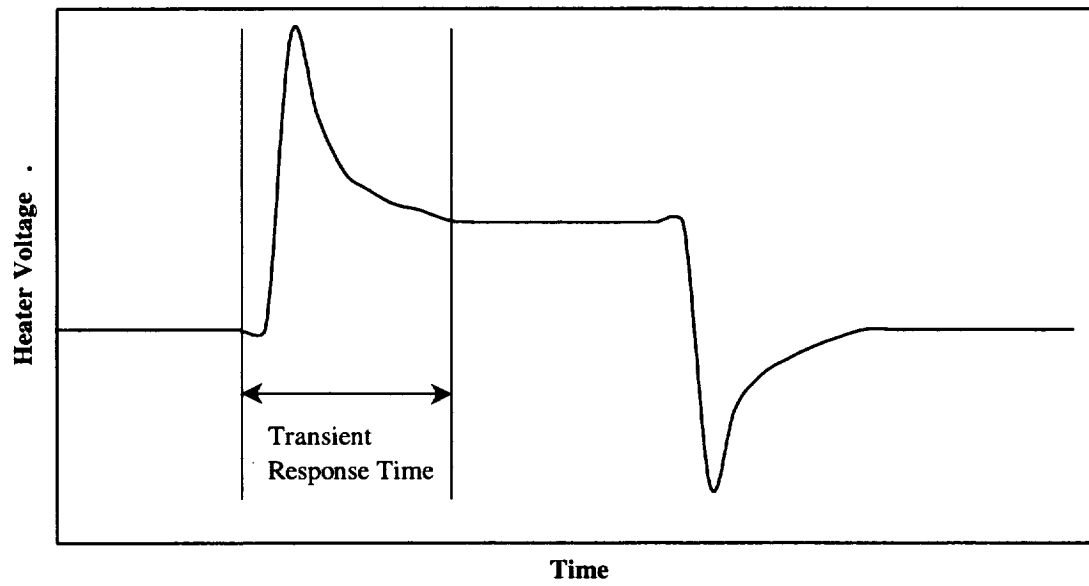


Figure 3.5: Transient response of feedback circuit

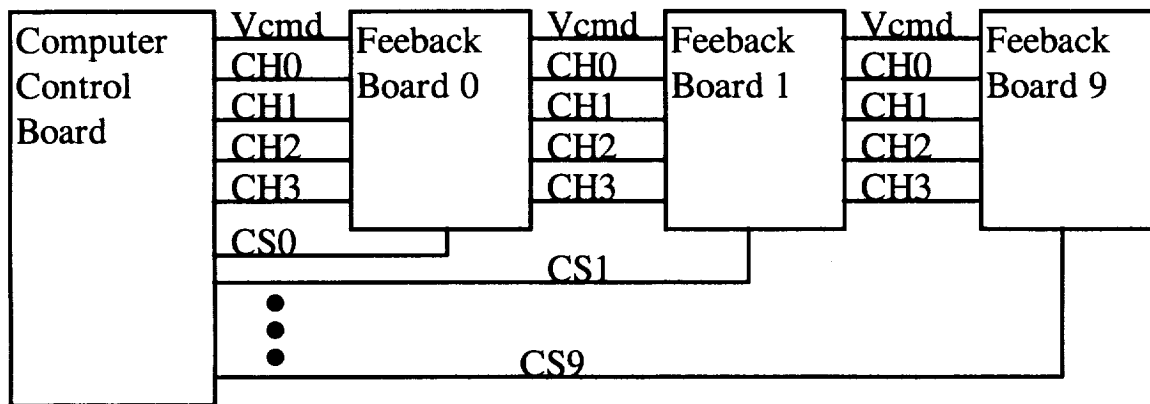


Figure 3.6: Computer control board multiplexing scheme

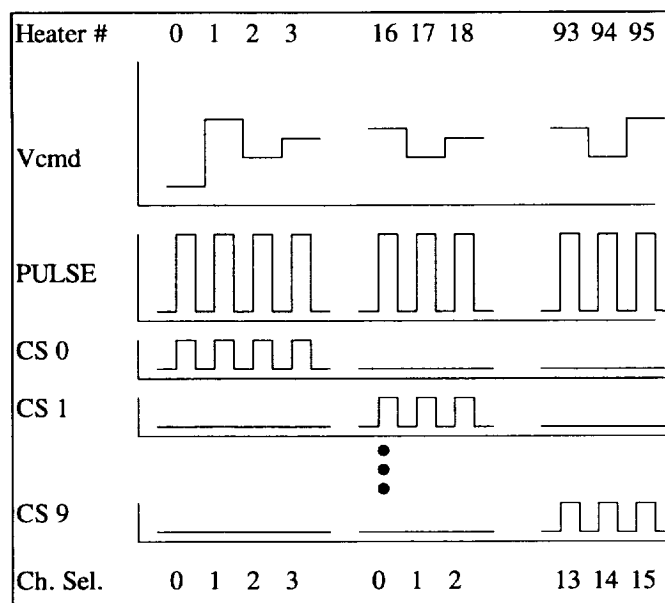


Figure 3.7: Timing for multiplexing method

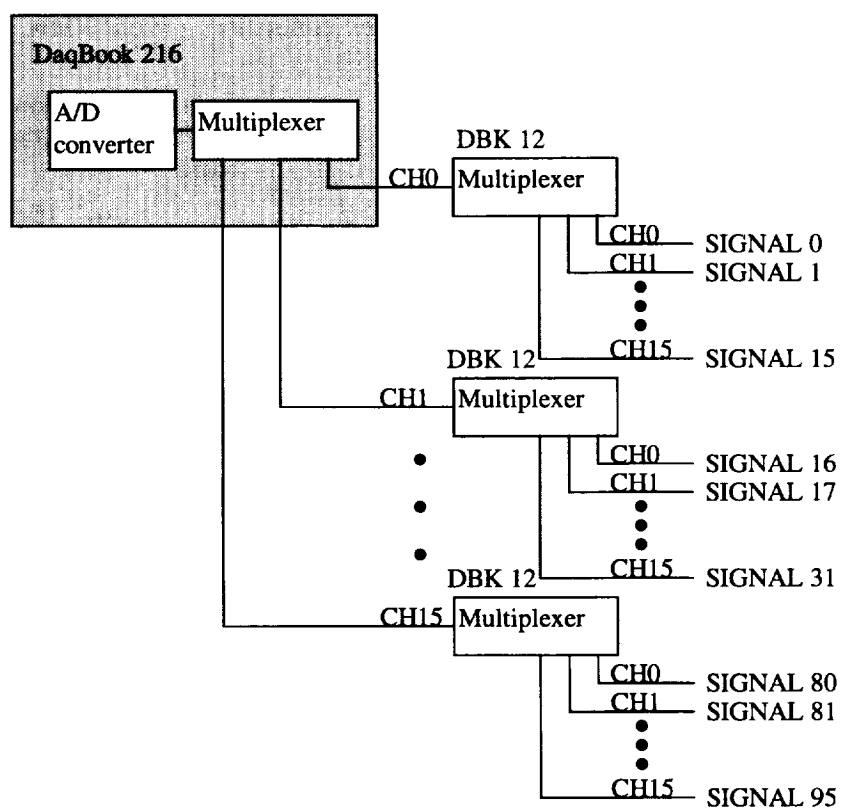


Figure 3.8: Daqbook Multiplexing Scheme

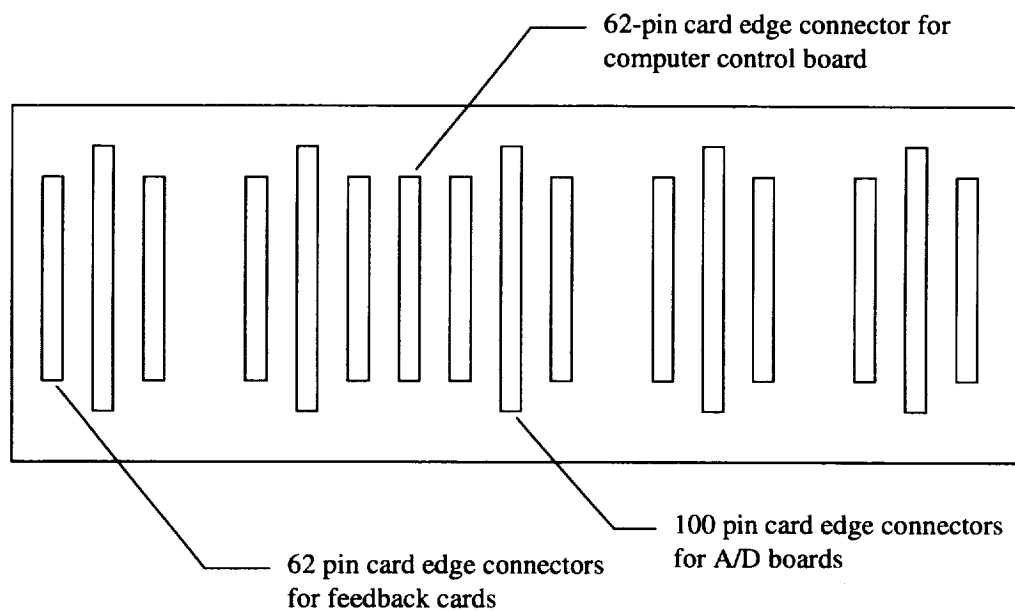


Figure 3.9: Arrangement of backplane connectors - Front View

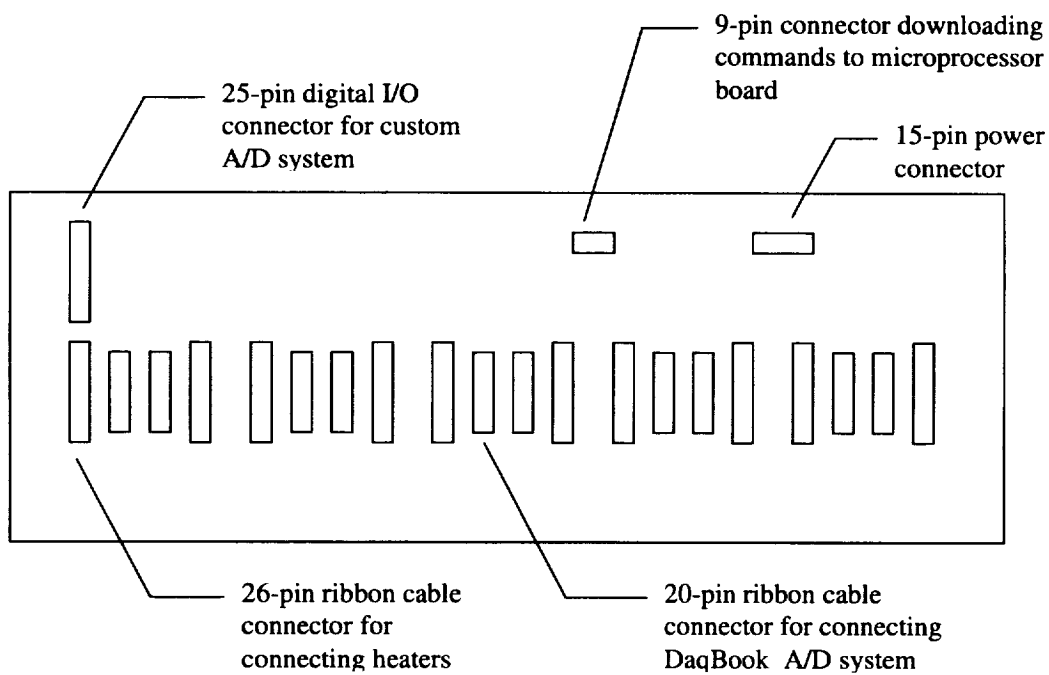


Figure 3.10: Arrangement of backplane connectors - Back View

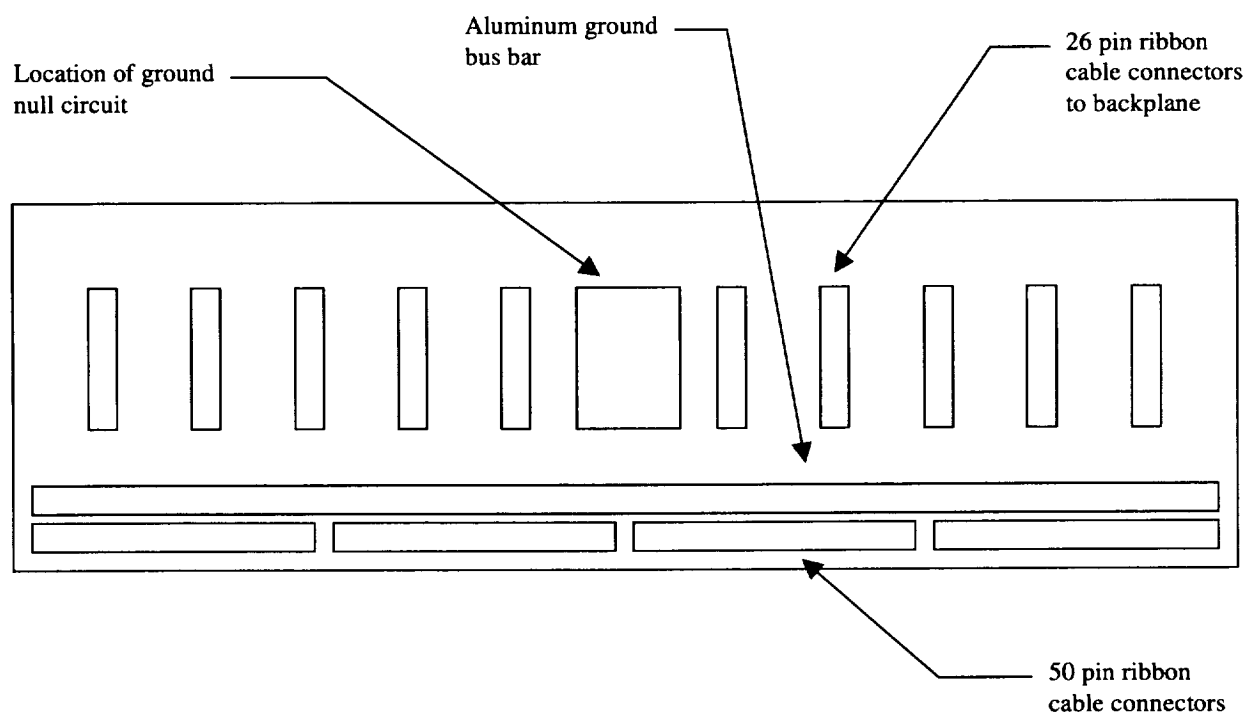


Figure 3.11: Decoding card layout

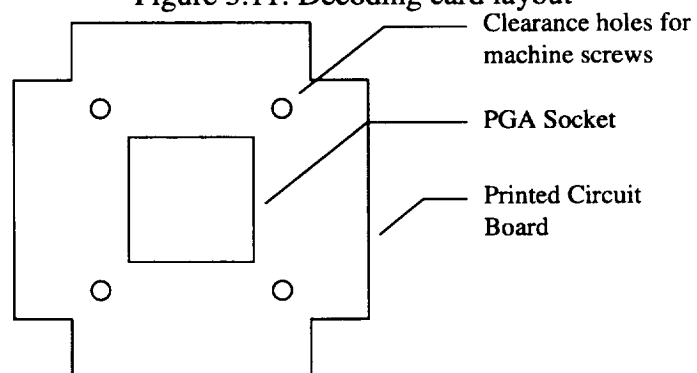


Figure 3.12: Printed circuit board to hold heater array

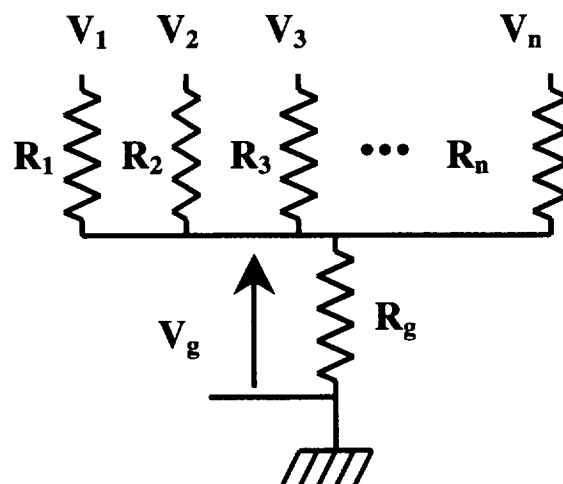


Figure 3.13: Common ground impedance

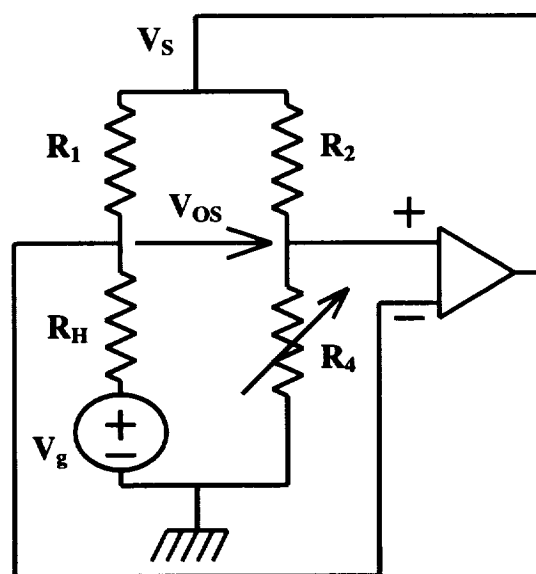


Figure 3.14: Effect of ground impedance in feedback circuit



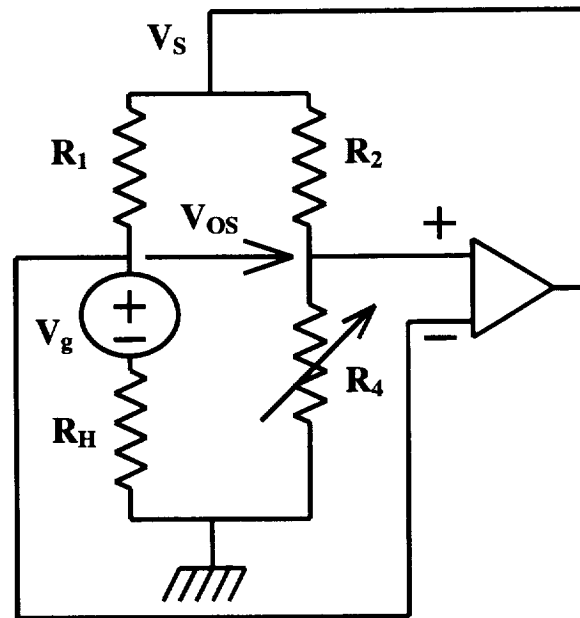


Figure 3.15: Effect of ground impedance in feedback circuit

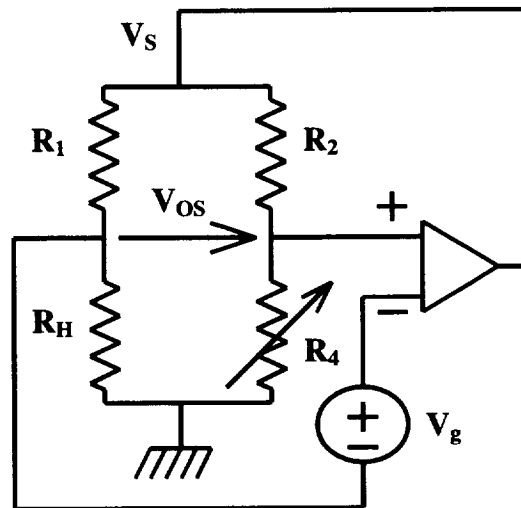


Figure 3.16: Effect of ground impedance in feedback circuit

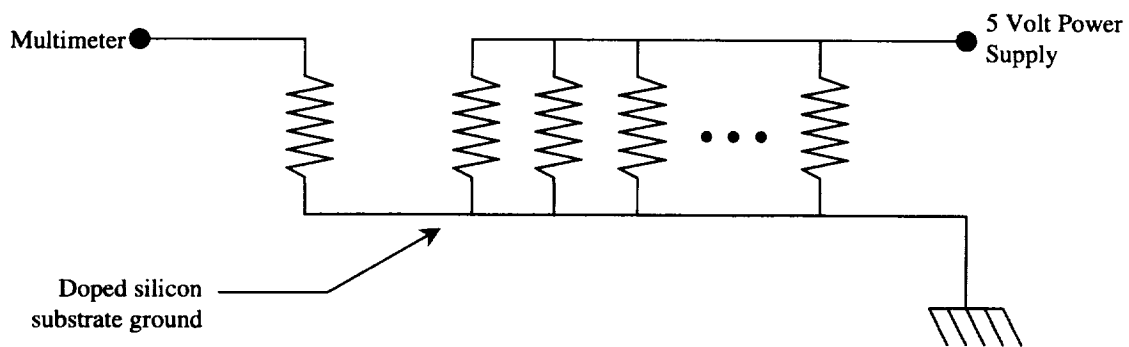


Figure 3.17: Schematic of ground potential experiment

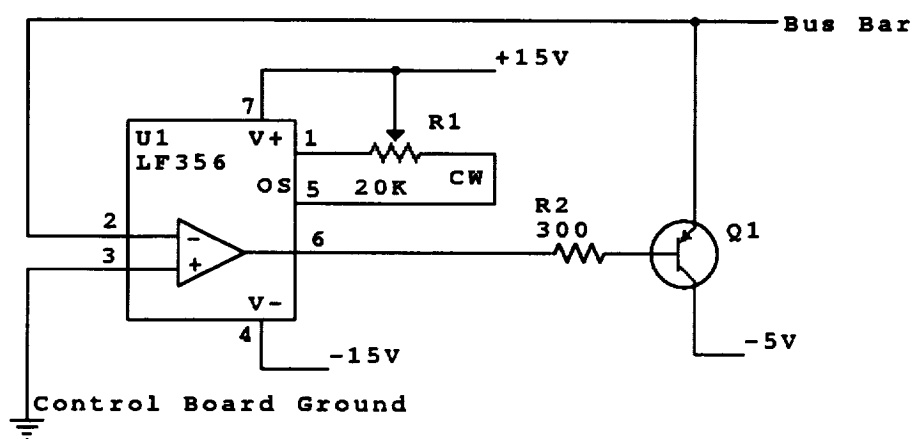


Figure 3.18: Schematic of ground null circuit

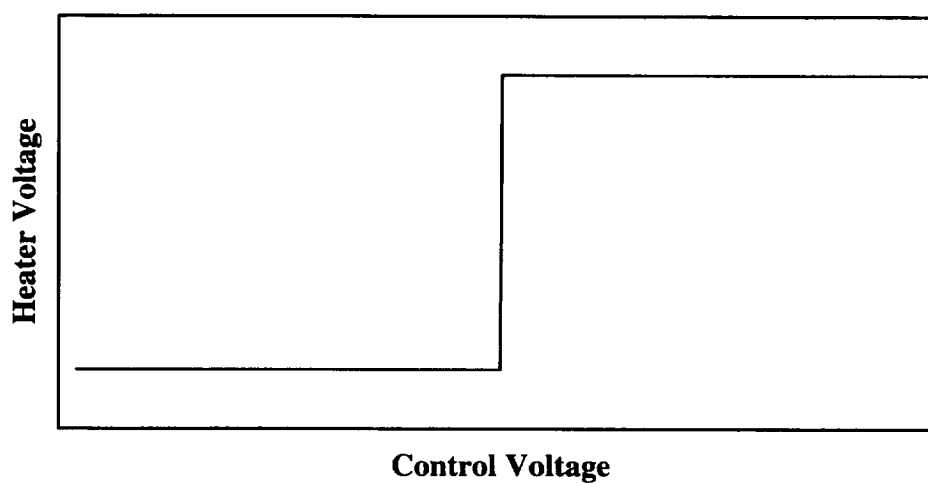


Figure 3.19: Ideal control-circuit response for constant-resistance

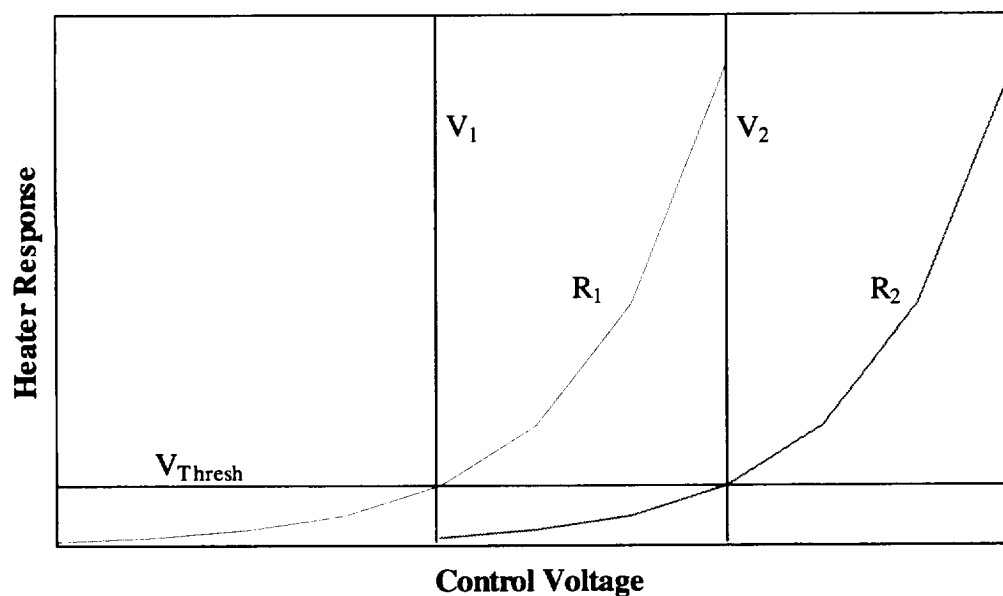


Figure 3.20: Heater Voltage vs. Control Voltage, Constant Temperature Lines.

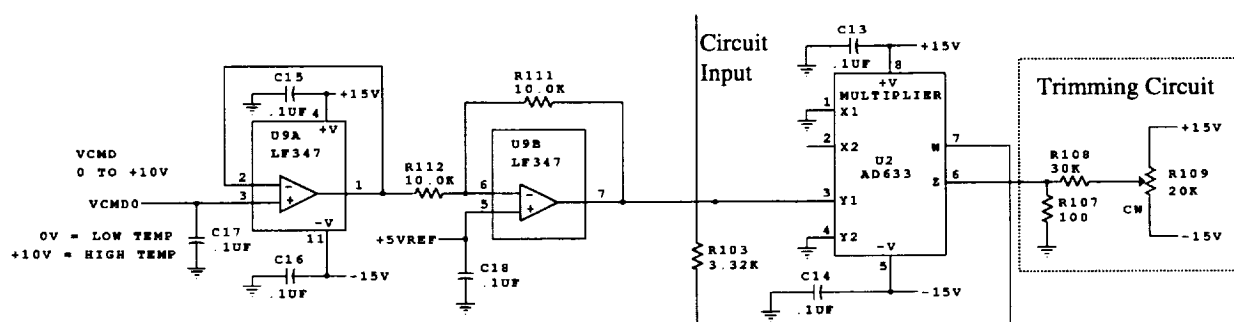


Figure 3.21: Circuit for correcting analog multiplier output offset

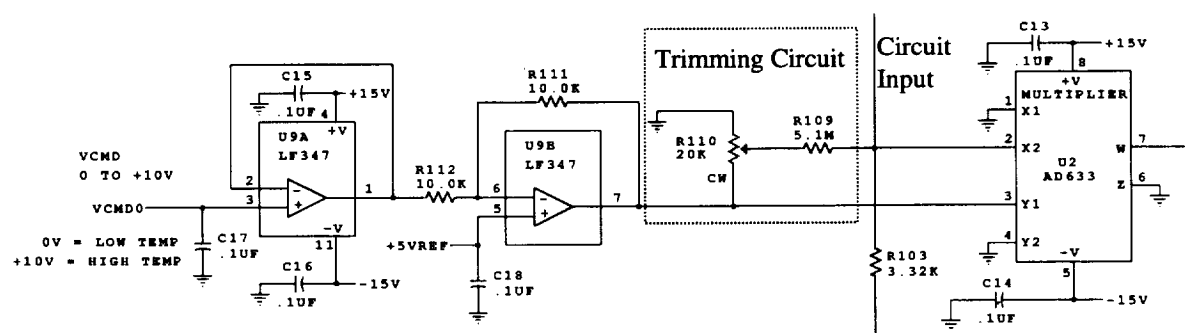


Figure 3.22: Alternative circuit for correcting analog multiplier output offset

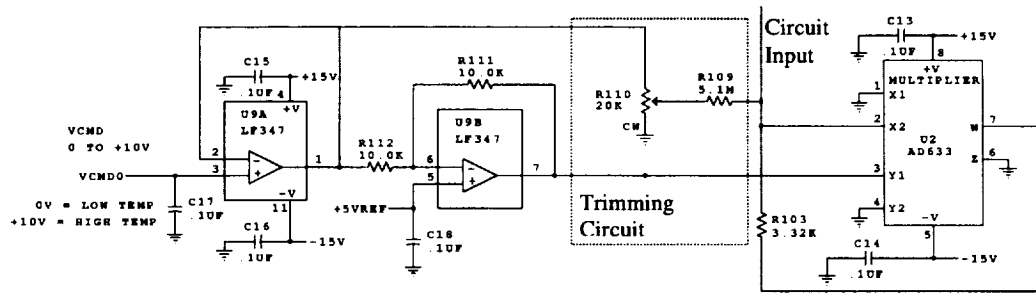


Figure 3.23: Modification to Figure 3.22 for correcting analog multiplier output offset

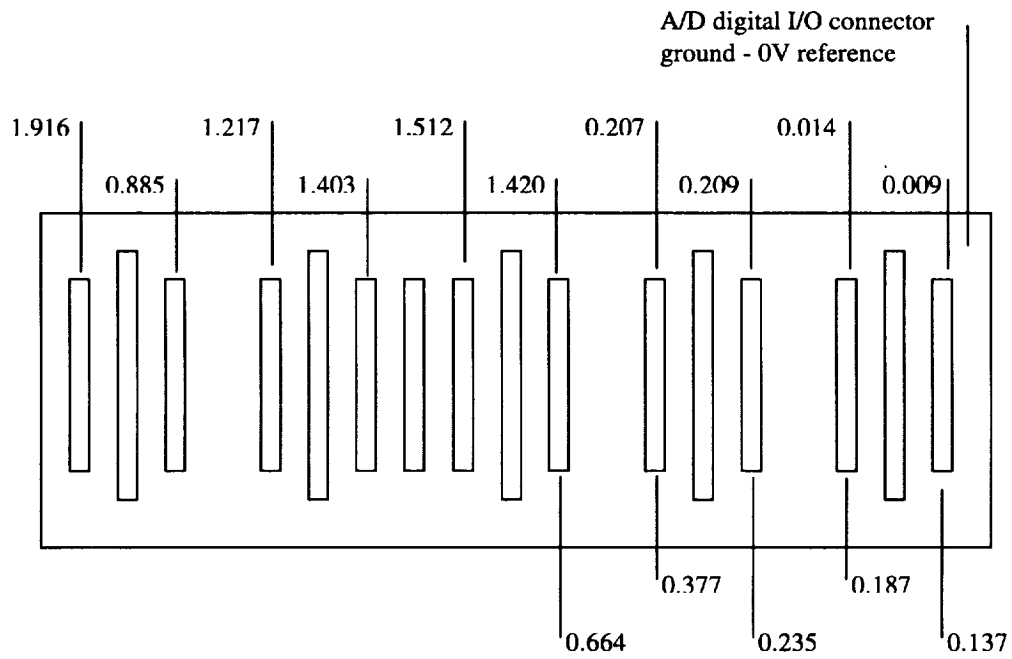


Figure 3.24: Ground voltage at various points on backplane board. Measurements are given in units of mV.

## CHAPTER 4: EXPERIMENTAL APPARATUS

The constant-temperature microscale heater array that has been developed requires a calibration apparatus that can control the calibration bath temperature and provide a high heat transfer coefficient to the surface of the heater. An experimental apparatus is required which can regulate the temperature and pressure of a working fluid in the chamber, and facilitate the removal of dissolved gasses from the working fluid.

### 4.1. CALIBRATION APPARATUS

A calibration apparatus was built to provide a temperature-controlled liquid bath in which the constant-temperature heater can be calibrated. Figure 4.1 shows a schematic of the apparatus. The apparatus consists of an insulated stainless-steel vessel, a heated temperature-controlled circulator, a pump which provides a jet of liquid directly onto the heater array surface to increase the forced convection heat transfer coefficient, and a lid to prevent the calibration fluid from splashing. Vegetable oil was used in the liquid bath because of its availability and high boiling point.

The accuracy of the temperature-controlled circulator was confirmed to within  $\pm 0.1$  C using a mercury thermometer. The circulator agitated the liquid bath, provided heating control to keep the temperature of the bath constant, and provided a readout for setpoint temperature and actual temperature. The bath temperature could be adjusted from about 25°C to about 100°C.

The liquid jet on the surface of the heater array was provided by a gear pump which pumped liquid through a 0.1 inch orifice above the heater array. The arrangement of the orifice relative to the heater array surface is shown in Figure 4.1.

Several tests were performed to determine an acceptable threshold voltage for the heater calibration. Section 3.6 discusses the meaning of the threshold voltage in greater detail. When the heater setpoint approaches the temperature of the liquid, the feedback circuit begins to supply a small amount of power to the heater. This power level is large enough that it can cause the heater temperature to rise several degrees above the calibration bath temperature. The threshold voltage is the heater voltage that cause the heater temperature to rise by an amount which falls within the desired calibration uncertainties. A precision power supply was used to supply a stable voltage, and a multimeter was used to measure the current through the heater and the voltage across it, and the resistance and temperature were determined from these values. Figure 4.2 shows a plot of the temperature rise vs. heater voltage, with and without an impinging jet. It can be seen from Figure 4.3, which shows the low voltage region of Figure 4.2, that using the impinging jet, the temperature rise is less than 1°C when the heater voltage is around 0.9V. This is an acceptable temperature rise for these preliminary experiments.

The computer software that controls the temperature of all the heaters allows the heaters to be calibrated automatically. The software uses a bisection method to find the value of  $V_{cmd}$  where the heater voltage reaches the threshold voltage. Appendix B gives a complete description of the calibration program.

## **4.2. CALIBRATION METHOD**

When the heater array is calibrated, the DaqBook data acquisition unit is used instead of the custom A/D system. The custom A/D system is capable of taking data from all the heaters at a very high sampling rate, but it is not practical to use it for calibration. The user must wait at least 1.6 seconds after triggering the custom A/D system to allow the A/D system to fill the input buffer, and then much longer to download a representative number of data points, even from one

heater. The Daqbook, however, allows the user to sample the voltage from a single heater and to download it into memory very quickly. This allows calibration to proceed more quickly.

The heater array is calibrated using the following procedure:

1. The orifice plate is attached to the PC board that holds the heater array so that the flow from the orifice is directed onto the heater array surface.
2. The heater array and orifice plate are placed in the liquid bath.
3. The circulator is turned on and the temperature setpoint is adjusted to the first calibration point.
4. The liquid bath temperature is allowed to stabilize near the setpoint.
5. The gear pump which provides flow through the orifice is turned on to full-speed.
6. The automated calibration routine on the PC is started, and a  $V_{cmd}$  value for each heater is obtained
7. The setpoint on the circulator is changed to the next calibration point, and the last three steps are repeated.

### 4.3. TEST CHAMBER

Experiments were performed in the test chamber shown in Figure 4.4. The apparatus consists of an aluminum test chamber which contains FC-72, and an aluminum compressed-air chamber which controls the pressure of the test chamber. A stainless-steel bellows is used to couple the pressure of the compressed-air chamber to the test chamber. The temperature and the pressure of the system can be adjusted to provide a wide range of saturated or sub-cooled conditions.

The temperature is controlled by an Omega brand temperature control unit. The unit is programmable for different control algorithms and thermocouple types. The default control

algorithm was used with a type-K thermocouple. The pressure is measured using an Omega pressure transducer with a polysilicon strain element.

The temperature controller was calibrated by placing the thermocouple sensor in a bath of known temperature and measuring the readout temperature. The unit was calibrated at three temperatures, and a second-order least-squares fit was used to interpolate between these temperatures. The curve was only slightly non-linear. Figure 4.5 shows the three temperature points that were used, and the best-fit line through those three points.

This temperature unit regulates the current through a set of Kapton heaters that are attached to the outside of the test chamber. The aluminum walls conduct heat from the Kapton heaters into the fluid, keep the fluid temperature regulated and maintain an even temperature around the test chamber because of the thickness and high thermal conductivity of the aluminum. Temperature stratification in the test chamber can be further reduced with the built-in stirrer. Pressure can be adjusted by opening valve  $V_1$  to allow compressed air into the air chamber, or opening valve  $V_2$  to draw down the pressure in the compressed air chamber.

Since the pressure and the temperature can be regulated, the system pressure and subcooling level can be varied in the experiments. The pressure can be varied from vacuum to 45 psia, and the liquid temperature can be varied from 25°C to about 100°C.

The liquid can be degassed by manipulating the valves  $V_1$  through  $V_5$  to vary the pressure and the temperature within the test chamber so that dissolved gasses are extracted from the liquid and vented from the test chamber. This procedure is described in greater detail in Chapter 5.

Lighting can be provided by quartz-halogen lamps operating on DC power. In this experiment, however, no photographic results are presented, so lighting was not critical.



It is sometimes important to note the amount of extension in the bellows. For instance, when the bellows begin to expand as the temperature was increased or the pressure was lowered, it indicates that a saturated conditions was obtained. The extension in the bellows could be checked by peering through a viewport in the side of the compressed air chamber and observing the position of the bellows on a graduated scale.

When the air chamber is being filled, it is necessary to keep the pressure from becoming too high. For this reason, a pressure regulator is used to limit the pressure at the inlet, and a pressure relief valve is used to vent the chamber if the pressure becomes too high.

When the test chamber is filled with FC-72, the liquid passes through a 2-micron filter to remove small particles. This is important so that these particles will not scatter light during photography, or interfere with the boiling mechanism.

#### 4.4. FIGURES

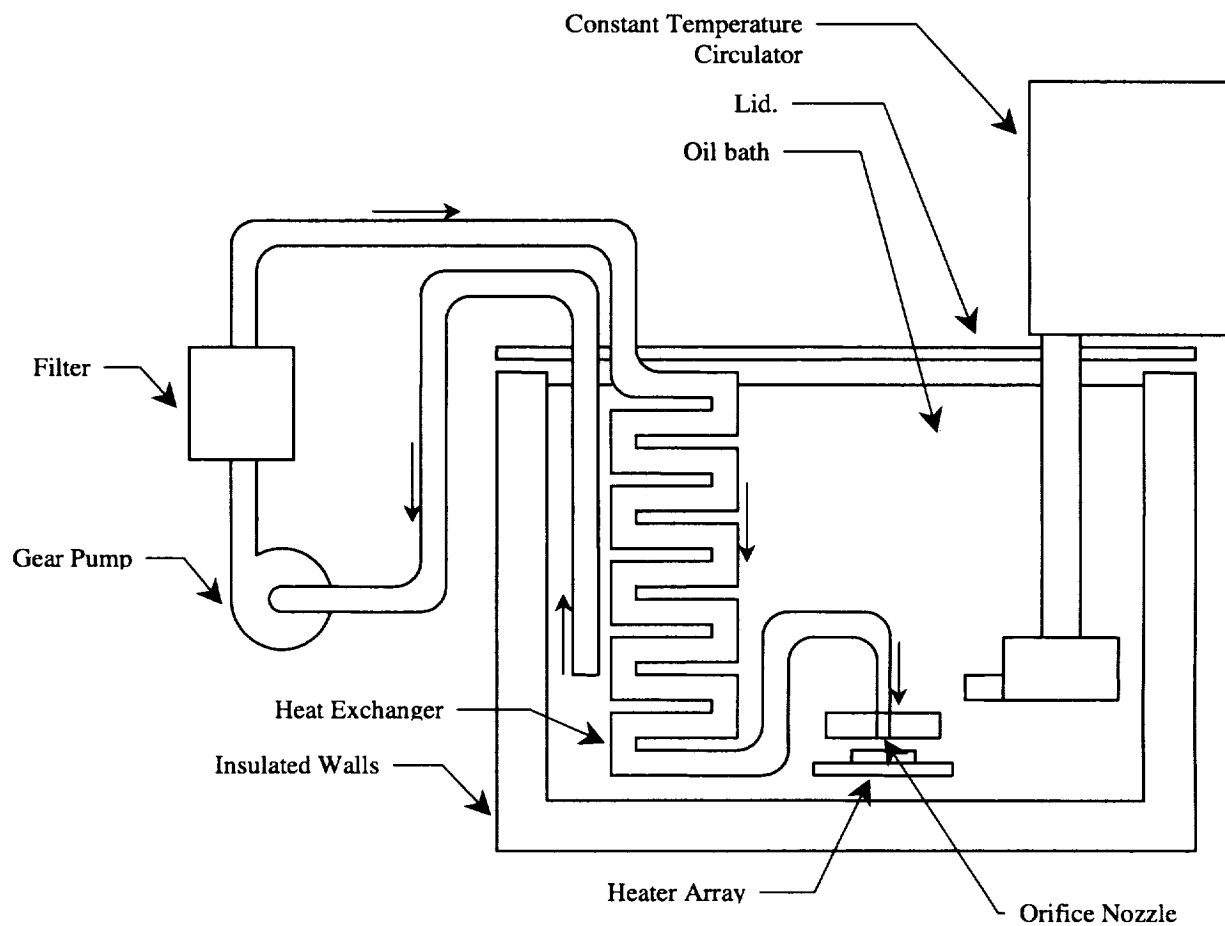


Figure 4.1: Calibration Apparatus

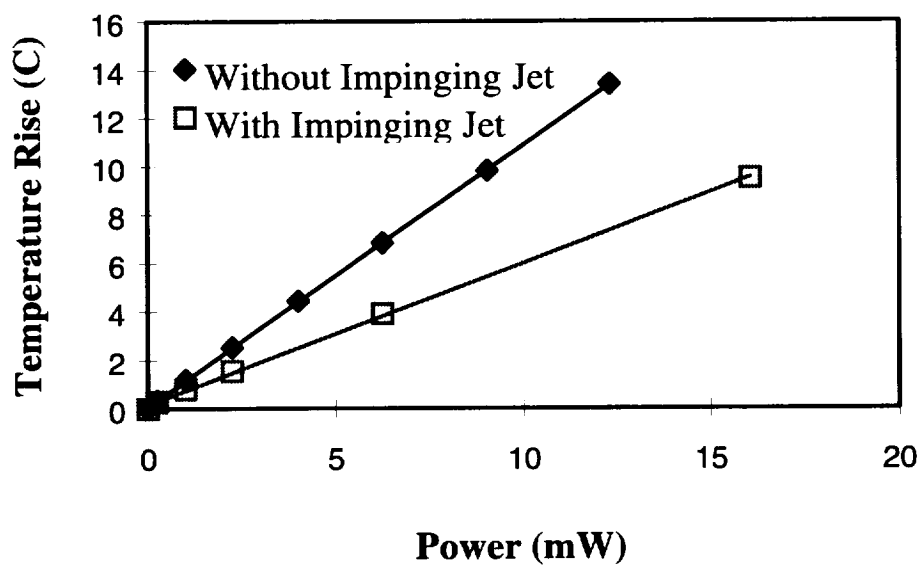


Figure 4.2: Heater temperature rise with and without impinging jet

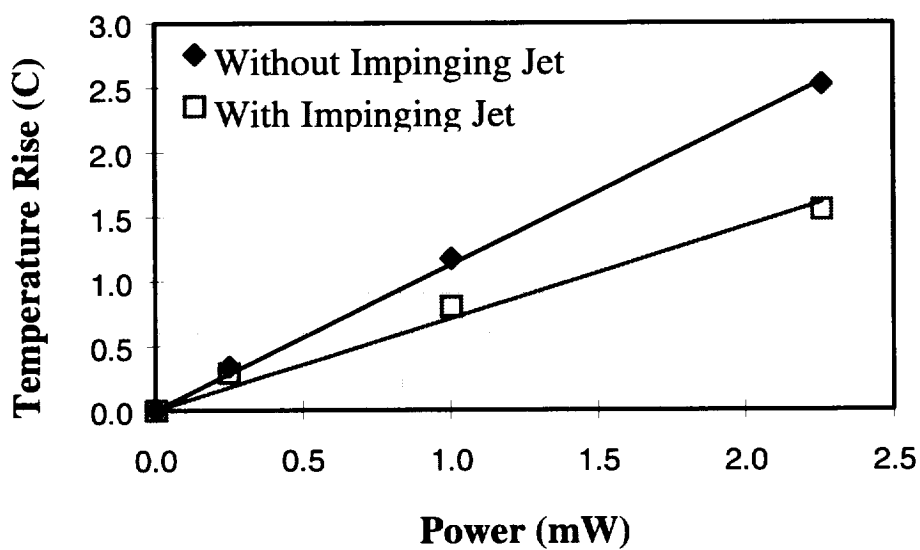


Figure 4.3: Heater temperature rise showing threshold voltage

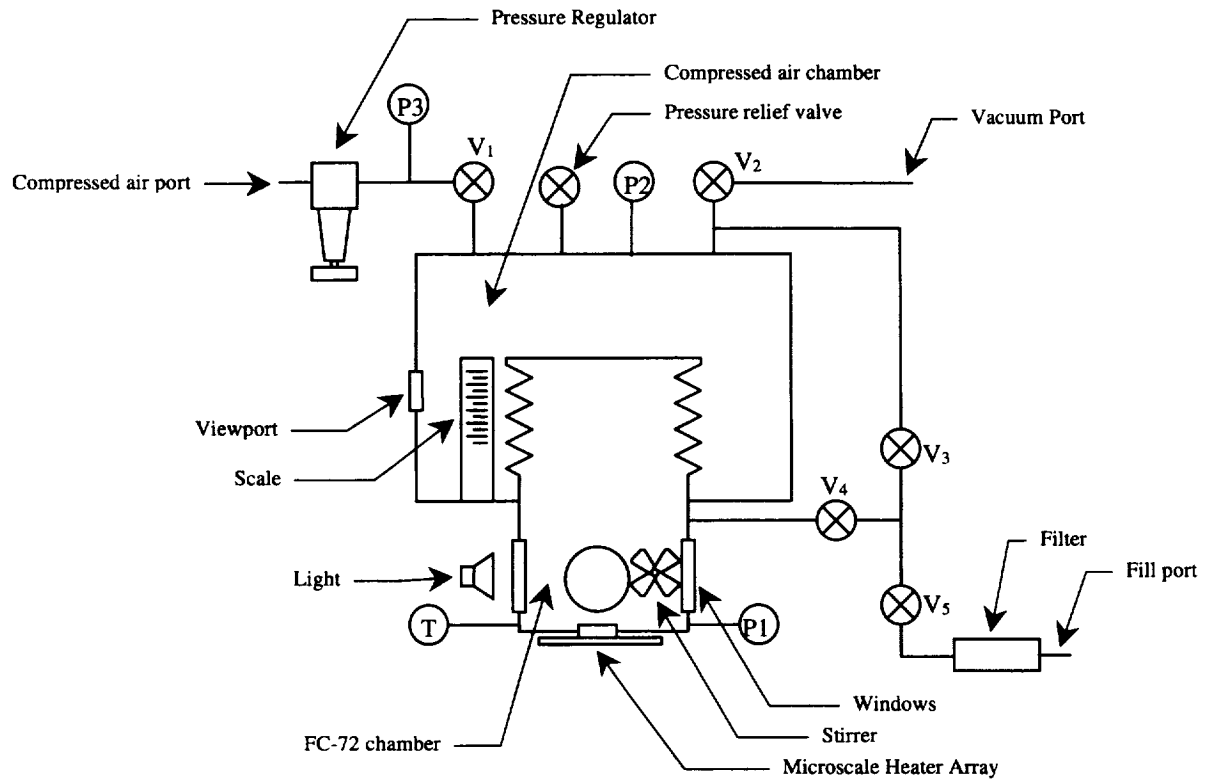


Figure 4.4: Experimental Apparatus

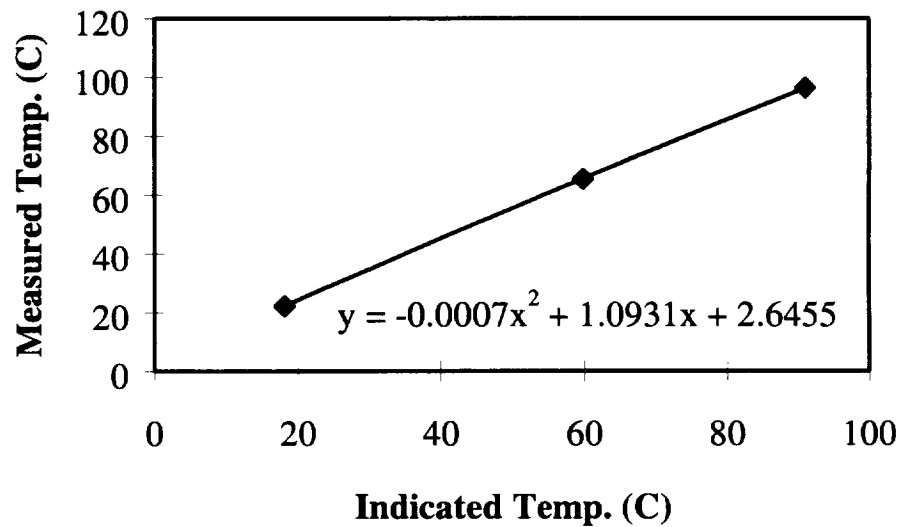


Figure 4.5: Temperature indicator calibration

## **CHAPTER 5: HEAT TRANSFER BEHAVIOR ON SMALL HORIZONTAL HEATERS DURING POOL BOILING OF FC-72**

This chapter presents some preliminary results that were obtained using the constant temperature heater array. Local measurements of wall heat flux during saturated pool boiling of FC-72 on a small heated area were made using an array of 96 temperature controlled heaters. The time resolved data was used to conditionally sample the heat flux according to whether or not boiling occurred on the surface, enabling the separation of the heat flux due to boiling from that due to natural convection or vapor contact. Significant variation in heat flux and boiling behavior were observed across the surface, indicating that data obtained from single point measurements may not be representative of average boiling behavior. The heat transfer from the edge heaters was observed to be much higher than that for the inner heaters above the critical temperature. The heat transfer during liquid contact in transition boiling was constant for a given wall superheat for the inner heaters, and was observed to decrease with increasing wall superheat. The scale of the individual heaters is approximately the same as that of the departing bubbles in nucleate boiling.

### **5.1. EXPERIMENTAL PROCEDURE**

#### **5.1.1. Degassing of fluid**

In order to minimize dissolved gas effects, the FC-72 liquid was degassed using the following procedure. After the test chamber was filled with gas saturated FC-72, the chamber was sealed and heated to the saturation temperature corresponding to atmospheric pressure in Denver, CO ( $T_{\text{sat}} = 52.6 \text{ }^{\circ}\text{C}$ , 0.85 atm). A vacuum pump was then used to draw down the bellows to increase the volume of the test chamber. Once steady state was reached (about 12

hours), the chamber was opened to atmosphere and the vapor quickly vented. This was repeated until the pressure within the test chamber reached the saturation pressure of FC-72 at the given bulk temperature. The final dissolved gas concentration in the liquid, determined using the chamber temperature and pressure and the properties of FC-72 (3M Fluorinert Manual, 1995), was less than  $1.5 \times 10^{-3}$  moles/mole.

### **5.1.2. Heater calibration**

The heater array was calibrated in the insulated, circulating constant temperature oil bath described in Chapter 4. The bath was held within 0.2 °C of the calibration temperature, while an impinging jet of oil onto the heater provided a high heat transfer coefficient. Heater array temperatures between these values were obtained by interpolation.

### **5.1.3. Data acquisition**

Both of the available data acquisition systems were used. Data was acquired using the Daqbook 216 from I/O Tech at a sampling rate of 20 Hz over 50 s, after allowing the heater array to remain at a set temperature for 10 minutes. The long time between data points was chosen because boiling on the surface was observed to change with time. Figure 5.1 shows how the array averaged heat flux changed after the wall superheat was decreased suddenly from 47.5 °C to 17.5 °C. Step changes in the wall heat transfer are observed. Visual observations of the heater showed that the steps in heat flux were associated with nucleation sites being reactivated as evidenced by new streams of bubbles appearing on the surface. The custom high-speed data acquisition system was also used to obtain time-resolved data at 2500 samples/sec from each heater for 0.5 seconds, in order to study time-dependent features of the boiling process.

#### **5.1.4. Data reduction**

The heat flux calculated from the voltage across the heater and the heater resistance ( $q_{\text{raw}}$ ) must be made to account for substrate conduction. The procedure used to determine the magnitude of this correction is explained below.

Wall heat flux vs. wall temperature were measured for each heater over the range  $T_w=50$  to  $100\text{ }^{\circ}\text{C}$  with the test chamber pressurized to 2.72 atm and with the bulk fluid at the saturation temperature corresponding to atmospheric pressure. The increased pressure effectively suppressed boiling on the heater array over this temperature range. The measured heat flux from a heater under these conditions represents the sum of natural convection and substrate conduction from that heater ( $q_{\text{nc+sc}}$ ). The magnitude of  $q_{\text{nc+sc}}$  compared to  $q_{\text{raw}}$  over the temperature range of interest is seen in Figure 5.2. Although  $q_{\text{nc+sc}}$  is relatively small compared to  $q_{\text{raw}}$  in the high nucleate boiling and CHF regimes, it becomes a substantial fraction of  $q_{\text{raw}}$  in the low nucleate boiling and transition boiling regimes. The average natural convection component over the entire heater ( $q_{\text{nc}}$ ) was calculated using the correlation of Lloyd and Moran (1974), (flat, upward facing, isothermal heaters), then subtracted from  $q_{\text{nc+sc}}$  to obtain the heat flux due to substrate conduction ( $q_{\text{sc}}$ ).  $q_{\text{sc}}$  for each individual heater was then subtracted from  $q_{\text{raw}}$  to obtain the heat flux due to boiling ( $q_{\text{b}}$ ). Because the heater array is held at constant temperature, the value of  $q_{\text{sc}}$  for each individual heater does not depend on the state of the fluid above the surface.  $q_{\text{nc}}$  was not subtracted from  $q_{\text{raw}}$  since natural convection does not exist once boiling occurs. Heat fluxes in the natural convection portion of the boiling curve are not presented because the above procedure simply results in the calculated value of  $q_{\text{nc}}$ .

### **5.1.5. Uncertainty analysis**

The circuit that allows the wall temperature to be adjusted exhibits offset voltages and nonlinearities which must be carefully compensated for. These compensations can still result in a certain amount of uncertainty in heater temperature since the offset voltages can drift with time. The temperature uncertainty due to this drift is estimated to be 0.4 to 0.6 °C. The calibration bath temperature uncertainty is small compared to the above uncertainties, and can be neglected. However, the value of the heater calibration could be as much as 0.7 °C too high because of self heating, since it is necessary to calibrate the heaters at a current level of close to 1 mA in order to reduce the uncertainty in the electrical measurements. An additional uncertainty arises in controlling the bulk fluid temperature, which was seen to vary during a ten hour run by about 0.5 °C. The final uncertainty in wall superheat was calculated to be 1.1 °C.

The uncertainty in  $q_b$  results from uncertainties in  $q_{raw}$ ,  $q_{nc+sc}$ ,  $q_{nc}$ , and  $q_{sc}$ . Uncertainties in  $q_{raw}$  and  $q_{nc+sc}$  arise from uncertainties in measured voltage across the heaters, the resistance of the heaters, and the heater array area. All three of these uncertainties are very small compared to the uncertainties that follow, and can be neglected. The uncertainty in  $q_{nc}$  was assigned the value of 100% since significant variations in  $q_{nc}$  can occur across the heater array. However, since  $q_{nc}$  represents only about 12% of  $q_{nc+sc}$ , this results in a relatively small uncertainty in  $q_{sc}$ . An additional uncertainty in  $q_{sc}$  results when boiling occurs on the surface since the heat transfer coefficient in the fluid surrounding the heater increases above that associated with natural convection, and this can change the value of  $q_{sc}$  calculated from the above procedure. To quantify this effect, a FLUENT simulation was performed in which the heat transfer coefficient on the substrate surrounding the heater was increased from 300 to 1000 W/m<sup>2</sup>-K (typical values of the natural convection heat transfer coefficient obtained from the above correlation varied



between 280-380 W/m<sup>2</sup>-K). Variations in  $q_{sc}$  of approximately 11% were observed. The uncertainties in  $q_{raw}$ ,  $q_{nc+sc}$ ,  $q_{nc}$ , and  $q_{sc}$  were combined according the methodology of Kline and McClintock (1953) to find the final uncertainty at each data point.

## 5.2. RESULTS

Before the formal experimental apparatus was completed, some preliminary, qualitative tests were performed to determine whether the heater array was capable of meeting the objectives of the research. These results are discussed in Section 5.2.1. . The remaining sections discuss the results that were obtained using the experimental technique explained in the previous sections.

### 5.2.1. Preliminary space-resolved, time-resolved results

For this preliminary experiment, the heater array was calibrated in an oil bath placed on a hot-plate/magnetic stirrer and heated to 100 °C. The heater was positioned facing upward in a 190 mm × 100 mm beaker containing a pool of liquid approximately 6 cm deep. The beaker was covered by a Teflon lid with slots to allow the power cables to pass through. The hot plate temperature was adjusted repeatedly and the bath allowed to come to thermal equilibrium until the desired calibration temperature was obtained. Bath temperature was measured with a type K thermocouple.

The preliminary test was performed in FC-72. The liquid was gas-saturated and at 25°C. The boiling point of FC-72 was measured and found to be 55°C in Denver, Colorado so that the liquid subcooling was 30°C. The heater was operated at 100°C . The liquid was not agitated during the test, and the heater was covered by approximately 1 cm of liquid.

In order to observe the local surface heat flux during pool boiling, the heaters in the array were set to 100°C and allowed to operate for several minutes. The Daqbook A/D system was

then used to digitize the heater voltage values. Each channel was digitized at a rate of 1 kHz, and data was collected for 0.5 s. All data presented here were taken with all 96 heaters, except for the 5 non-functional heaters shown in Figure 5.3. These 5 heaters were not functioning because of flaws in the wire-bonding connections that attach the heater array to the PGA package.

Figure 5.4 shows the time response of a single heater at the center of the array with boiling taking place on the heater array surface. In this figure, the heat transfer coefficient is defined as  $h = q'' / (T_{wall} - T_{sat})$ . The periodic growth and departure of several bubbles can be seen as a periodic change in the heat transfer coefficient. This data was sampled at a rate of 50 kHz, or about 500 heat flux maps per second. The heater is seen to follow the transients in wall heat transfer due to boiling with high temporal resolution.

The heat transfer coefficient for the 64 center heaters was averaged over the surface at 1 ms intervals and plotted vs. time. The results are shown in Figure 5.5 for one growth and departure cycle. Surface plots of the heat transfer coefficient from the center 64 heaters representing a 2 mm × 2 mm area for one bubble cycle are shown in Figure 5.6. Fifteen time steps are shown at 2 ms intervals. The times on Figure 5.6a-Figure 5.6o correspond to the times on the X-axis of Figure 5.5.

It could be seen from visual observation of the heater array surface that multiple bubbles were forming on the surface, coalescing, and then departing as a single bubble. This observation is confirmed by the surface plots of the heat flux. In Figure 5.6, the two low heat transfer regions in front and back, which are believed to represent areas covered by vapor bubbles, are beginning to coalesce. In Figure 5.6b-Figure 5.6h, these two bubbles completely coalesce into a single bubble which covers most of the surface. In Figure 5.6i-Figure 5.6n, the bubble departs from the

center of the heater, and the low-heat-transfer region in the center disappears completely as subcooled liquid rushes in. In Figure 5.6n and Figure 5.6o, three new low-heat transfer regions are seen to be growing and preparing to coalesce.

According to Figure 5.5, the maximum heat transfer coefficient occurs at 25 ms, corresponding to Figure 5.6m, and appears to be during bubble departure. Since this experiment was performed in a highly subcooled liquid, it is felt these results indicate that the dominant heat transfer mechanism in subcooled boiling is convection/conduction in the single-phase liquid, as other researchers have suggested. This is in contrast to the dominance of microlayer evaporation heat transfer in saturated pool boiling. Flow visualization using a high-speed digital video camera will be performed to confirm these results. Heat transfer coefficients up to  $28,000 \text{ W/m}^2\text{-K}$  were observed during the bubble departure cycle. The bubble departure frequency was approximately 30 Hz.

Preliminary results proved the feasibility of using a microscale heater for making local heat transfer coefficient measurements beneath a growing and departing bubble. The control system was able to make temporally and spatially resolved heat transfer coefficient measurements with unprecedented detail.

### **5.2.2. Spatially averaged, time averaged boiling curve**

For the remaining results, data was taken over a ten hour period in which the wall temperature was increased and decreased twice between  $65^\circ\text{C}$  and  $100^\circ\text{C}$  in  $2.5^\circ\text{C}$  increments. The bulk fluid was slightly subcooled by about  $1.5^\circ\text{C}$ . Two of the heaters near the edge of the array were not functional, as shown in Figure 5.7. The apparatus shown in Figure 5.8 was used for this series of experiments. The boiling curves for all four runs are shown on Figure 5.9. The boiling curves are seen to be remarkably repeatable with time, although a small hysteresis is

observed between the increasing and decreasing temperature runs. A hysteresis has been observed by other investigators (Ungar and Eichhorn, 1996, Rajab and Winterton, 1990). The source of this hysteresis is currently under investigation. The CHF for all cases is similar, however, and is seen to be somewhat higher than that obtained from a correlation for small vertical heaters of short width (Park and Bergles, 1988). This is not surprising since they operated their heaters in a constant heat flux mode, in contrast to the constant temperature boundary condition of the current array. Park and Bergles (1988) did observe that CHF increased for heaters with higher thermal conductivity, which is consistent with the observed trends. No hysteresis associated with boiling incipience was observed due to a flaw in the software -- whenever the temperature of the heaters is changed, the heater temperatures are set to random temperatures for about 1 second, causing some heaters to shut down while others are set to very high temperatures. Boiling therefore occurs somewhere on the heater every time the array temperature is changed.

### **5.2.3. Spatially resolved, time averaged RMS**

The root-mean-square (RMS) of the heat flux data is a means of characterizing the magnitude of the heat transfer variations on the surface of the heater. It may not truly reflect the probability distribution of the signal, since the heat transfer is not necessarily Gaussian, but it does give a first indication of how the average amplitude of the signal changes with surface temperature. Two methods were used to calculate the RMS of the heat flux. The first method is referred to as *spatially resolved* RMS. In this method, the RMS heat flux values were calculated for each heater element in the array, then averaged to obtain the average magnitude of the local RMS heat flux variation of each individual heater. The result is a quantity that characterizes the magnitude of the local heat flux variations during boiling on a constant-temperature surface. In

the second method, referred to as spatially *averaged* RMS, the spatially averaged heat flux at many times are calculated, from which the RMS value are computed. It must be remembered that local changes in heat flux that are on a smaller scale than one of the microscale heater elements cannot be measured accurately. The size of bubbles departing from the surface during low nucleate boiling was visually observed to be approximately the size of the individual heaters, so local heat flux during the early stages of bubble growth cannot be measured.

The spatially resolved RMS is represented in Figure 5.9 by a pair of lines which bound the average heat flux values. It is seen that the heat flux variations are on the order of the array averaged heat flux. Figure 5.10 shows the spatially averaged RMS values compared to the spatially resolved RMS values. The spatially resolved values are seen to be three to six times larger than the spatially averaged values. This is analogous to the temperature variations that are seen on a constant heat flux surface. For instance, the numerical work of Unal and Pasamehmetoglu (1994) predicts significant local temperature variations, but a nearly constant spatially averaged temperature due to the averaging of many out-of-phase local variations. Kenning (1992) observed local wall superheat variations of up to 150% of the mean value in nucleate boiling.

The difference in magnitude between spatially resolved and spatially averaged heat flux variations shows that techniques which use spatially averaged heat flux values to draw conclusions about small-scale heat transfer processes may need to be reexamined. For instance, Haramura and Takeno (1997) used the surface averaged heat flux measured from a small constant temperature surface to predict macrolayer thickness. In such experiments, the local heat balance may differ significantly in magnitude from the average heat balance.

Figure 5.11 shows surface plots of the locally resolved, time averaged heat flux values and locally resolved RMS values at three points on the boiling curve. One can see from Figure 5.11a that the peaks and valleys of the heat flux and RMS values tend to occur in the same regions of the heater. This provides strong evidence that the regions of high heat flux and high heat flux variation are areas where nucleation occurs. Figure 5.11b shows that the heat flux and RMS variation magnitudes do not vary significantly across the surface of the heater at CHF. This may indicate that nucleation is occurring evenly over the entire surface. The large low-spot on the rear left of the heater is due to the non-functional heaters shown in Figure 5.7.

Figure 5.11c shows that most of the heat transfer in transition boiling on small heaters occurs near the edge of the heater, yet there are very large variations in heat transfer over the entire heater. In fact, one can see from Figure 5.11c that the RMS variation at the center of the heater is larger than the corresponding heat flux. This is not possible if the RMS variations occur symmetrically about the mean, because it would result in negative heat flux values. Figure 5.12 is a plot of heat flux with time for a heater in the center of the array. The plot shows that the heat flux remains low except for short periods where it becomes very high -- up to  $60 \text{ W/cm}^2$ . The result is that the RMS is larger than the average without the heat flux becoming negative in value. This also shows that variations in heat flux can be much larger than the average heat flux at a given point on the surface.

#### **5.2.4. Spatially averaged, time resolved results**

Spatially averaged, time resolved heat flux is shown in Figure 5.13. Not surprisingly, the signal during nucleate boiling ( $\Delta T_{\text{sat}}=25 \text{ }^\circ\text{C}$ ) is seen to be quite random since the average heat flux over the surface is the result of boiling from many individual sites on the surface that do not necessarily occur in phase. A FFT of this signal showed no strong peaks. At CHF

( $\Delta T_{\text{sat}}=35^{\circ}\text{C}$ ), the heat flux begins to become dominated by the large vapor mass that occasionally departs the surface, and a quasi-periodic signal is observed to appear. A FFT indicated the emergence of several distinct peaks at approximately 26 and 52 Hz. In transition boiling, the secondary peaks become more pronounced and the overall heat flux level decreases.

#### **5.2.5. Time resolved data from individual heaters**

Seen on Figure 5.14 are time resolved heat flux traces for heater 18 in the array at various wall superheats. Consider first the heat flux trace for low nucleate boiling ( $\Delta T_{\text{sat}}=22^{\circ}\text{C}$ , Figure 5.14a). Examples of regions where natural convection, enhanced convection, and boiling are thought to occur on the surface are indicated. Nucleate boiling is assumed to occur when large variations in heat flux are seen. Natural convection is assumed to be characterized by low heat flux levels along with small variations in heat transfer. Enhanced convection is assumed to occur when the heat flux level is higher than occurs for natural convection, but without the large variations in heat flux characteristic of nucleate boiling, and could be heat transfer due to bubble induced liquid motion. The categorization of nucleate boiling and natural convection are based on observations of the heat flux variation on the surface and correlating them to the visual observations of boiling on the surface. For example, when large variations in heat flux at relatively high levels are observed on a particular heater, a stream of bubbles is seen from that heater location. The categorization of enhanced convection, however, is somewhat murky, since there is no method of distinguishing it from natural convection other than by the heat flux level.

The heat flux trace at a slightly higher wall temperature ( $\Delta T_{\text{sat}}=25^{\circ}\text{C}$ ) consisted almost entirely of nucleate boiling. The average heat flux level at this temperature was significantly higher than at lower temperatures, indicating that the increase in heat transfer from the surface is

partly due to increased heat transfer from individual heaters, and not just due to the activation of additional nucleation sites on the surface, or a higher bubble departure frequency.

At CHF ( $\Delta T_{\text{sat}}=35\text{ }^{\circ}\text{C}$ , Figure 5.14b), regions of low heat transfer are observed in the heat flux traces. It is felt that these represent the heat transfer when vapor covers the surface. Visual confirmation of this is not available at this time, but it is implausible that these low heat flux regions would represent natural convection due to the high level of bubble activity on the surface. The existence of vapor on the surface during CHF has also been observed by other researchers (Lee, et al., 1985, Kalinin, et al., 1987, Alem Rajabi and Winterton, 1988a, and Nishio, et al., 1997).

Within the transition boiling region (Figure 5.14c), the heat flux traces indicate that vapor covers the heaters for an increasingly larger fraction of time. The shape of the peaks is interesting. Typically, a sharp increase in heat flux is initially seen followed by a rapid decay and a sharp drop to the vapor state. Similar behavior was observed by Chen and Hsu (1995) who obtained transient wall temperature and heat flux measurements during liquid contact on a superheated surface. They observed that the wall heat flux jumped to a high level almost immediately upon liquid contact, but then decayed with time until the droplet evaporated or left the surface.

### **5.2.6. Spatially resolved, time averaged data**

Boiling curves generated for "rings" of heaters are shown on Figure 5.15. In the nucleate boiling region, all the heater rings have similar boiling curves. The scatter in the data for a given wall superheat is due to boiling being initiated at random sites on the surface. Note that CHF for the inner heaters (Rings 1-4) occurs at a lower wall superheat ( $30\text{-}32\text{ }^{\circ}\text{C}$ ) than CHF for the entire array (about  $35\text{ }^{\circ}\text{C}$ ). At a wall superheat of  $35\text{ }^{\circ}\text{C}$ , transition boiling occurs on the inner heaters.



The boiling curves begin to deviate from one another above a wall superheat of 32 °C. In transition boiling, the boiling curves for Rings 1 and 2 are very similar, and the minimum heat flux point is almost reached at the highest wall temperature. The boiling curves for Rings 3 and 4 are somewhat higher than those for the inner rings, while the curve for Ring 5 shows no decrease in heat flux over the wall superheats tested. In fact, the heat flux from the edge heaters (Ring 5) is seen to increase with wall superheat to a level about 150% above the CHF for the inner heaters at the maximum superheat. The explanation for this, of course, is that the edge heaters can be supplied with liquid from the side, while the liquid supply to heaters in Rings 1-4 is cut off by bubbles generated on the surface.

#### **5.2.7. Conditional sampling**

In order to conditionally sample the heat flux only when boiling occurs on the surface, a *boiling function* was generated. This function is a bimodal signal that is set to HIGH when boiling or enhanced convection occurs on the surface, and LOW otherwise. Enhanced convection was counted as HIGH when computing the boiling function since it represents additional heat transfer due to bubble motion on the surface. The *boiling fraction* is defined as the time average of the boiling function, and represents the time fraction that boiling or enhanced convection occurs on the surface. Within transition boiling, where the surface is alternately wetted by liquid and vapor, the boiling fraction related to void fraction according to

$$\text{Boiling Fraction} = 1 - (\text{Void Fraction})$$

A schematic of the process by which the boiling function is generated is shown in Figure 5.16. The time resolved signal is processed according to three criteria. Channel C is a simple level detector--if the heat flux is higher than what would be expected during natural or forced convection or when vapor covers the surface, then Channel C is set to HIGH. Simply relying on

this one channel, however, does not enable a clear discrimination of the boiling signal. Channel A assumes that the time derivative of the heat flux signal is large when boiling occurs on the surface, and is set to HIGH when the rectified first derivative exceed a certain threshold value. Channel B computes the rectified second derivative of the heat flux signal and compares it to a threshold value--the output is set to HIGH if the threshold value is exceeded. Channels A and B are used to solve the problem of zero crossing--the first derivative unavoidable falls below the threshold value as it changes sign, resulting in Channel B occasionally being falsely declared LOW. The zero crossing problem can be eliminated by monitoring Channels A and B simultaneously--both channels have crossing dropouts, but the zero crossings of the two channels do not coincide in time since the second derivative is zero when the first derivative is maximum and vice versa. The OR gate sets the boiling function to HIGH if the output from any of the three channels is HIGH. The three threshold values used produced acceptable boiling functions across the heater array at a given superheat, as well as for a given heater at various superheats. Examples of the performance of the above criteria in generating the boiling function for a given heater at various temperatures is shown on Figure 5.14a-c.

#### **5.2.8. Boiling fraction**

The time average of the boiling function for each heater in the array was calculated, and these values were averaged over heaters within a particular ring. This yields the boiling fraction for a particular ring, i.e., the fraction of time the heaters in the ring see boiling on the surface. A plot of the boiling fraction vs. wall superheat is shown on Figure 5.17. The boiling fraction is seen to reach a value close to unity at a wall superheat of about 25 °C, well before the temperature corresponding to CHF for all rings. The boiling fraction at CHF ( $\Delta T_{\text{sat}}=35$  °C) is observed to be lower than unity for the inner heaters, indicating the occurrence of vapor patches

on the surface. This is consistent with the data shown on Figure 5.15, which shows that the transition boiling region has been entered at this wall superheat for the inner rings of heaters. The boiling fraction for Rings 1-4 decreases quite rapidly after CHF. At the highest wall superheat, the innermost heaters see boiling only 6% of the time, while boiling is nearly continuous on the outer ring of heaters.

A monotonic increase in the boiling fraction with ring number is seen at any given superheat in transition boiling. This agrees with the observed bubble dynamics and wall heat transfer pattern. Within transition boiling, the heaters were rewet as the large vapor bubbles left the surface. The liquid front was observed to move from the outside of the heater array towards the inside heaters, then move outward again as the vapor bubble formed once again above the heater array. The heaters towards the outside of the array thus experience boiling a larger fraction of the time than the heaters toward the center of the array. This monotonic increase with boiling fraction during nucleate boiling is not seen since boiling can initiate anywhere on the surface.

#### **5.2.9. Conditionally sampled heat flux**

The time resolved heat flux data for each heater was conditionally sampled according to the boiling function to obtain the heat flux only when boiling or enhanced convection occurred on the surface, and averaged over the heaters within a particular ring. This results in the average heat transfer for a particular ring that is uninfluenced by heat flux during vapor contact or natural convection. This quantity is referred to as the *boiling heat flux*. The results are shown in Figure 5.18. It is seen that if one excludes the edge heaters (Ring 5), the boiling heat flux more or less collapses onto a single curve for all wall superheats, implying that the heat flux at a given wall superheat during boiling is constant over the heater surface. For example, the heat flux during

boiling at  $\Delta T_{\text{sat}}=47^\circ\text{C}$  for Ring 1 is similar to that for Ring 4, even though Ring 1 sees boiling only 6% of the time while Ring 4 sees boiling 55% of the time.

During transition boiling, the boiling heat flux for the inner heaters is observed to decrease with increasing temperature. The reason for this is currently not known, but it may be a result of the shorter liquid contact time and/or the establishment of a vapor layer underneath the liquid. Similar trends have been seen by other researchers (Marquardt and Auracher, 1990, Alem Rajabi and Winterton, 1988a,b, and Chen and Hsu, 1995). The current data indicates that some of the models used to predict transition boiling heat flux may be in error. Often, the heat flux during transition boiling is modelled as

$$q_{tr} = q_{CHF}(F) + q_{MHF}(1 - F)$$

where  $F$  is the fraction of time the surface is covered with liquid. A variation on this model is to extend the nucleate boiling and film boiling curves, and weight these values at a given wall temperature on  $F$  to find the transition boiling heat flux. It is seen from the current data that heat transfer during liquid contact does not remain constant nor does it increase in the transition boiling region as the above two models would suggest.

### 5.3. CONCLUSIONS

1. The array averaged heat flux was seen to vary quasi-periodically above CHF.
2. The inner heaters reach CHF at lower wall superheats than that for the array averaged heat flux.
3. Significant variations in boiling fraction occur over the surface of small heaters during nucleate and transition boiling, indicating that point measurements of heat flux, temperature, or void fraction may not be representative of average boiling behavior.

4. Vapor patches at CHF and during transition boiling were observed to move with time, and were related to the bubble dynamics above the heater.
5. Heat transfer during liquid contact in transition boiling was constant for a given wall superheat for the inner heaters, and was observed to decrease with increasing wall superheat.
6. The heat flux for the edge heaters was observed to increase continuously with wall superheat, reaching heat flux levels much higher than the maximum heat flux for the inner heaters.

## 5.4. FIGURES

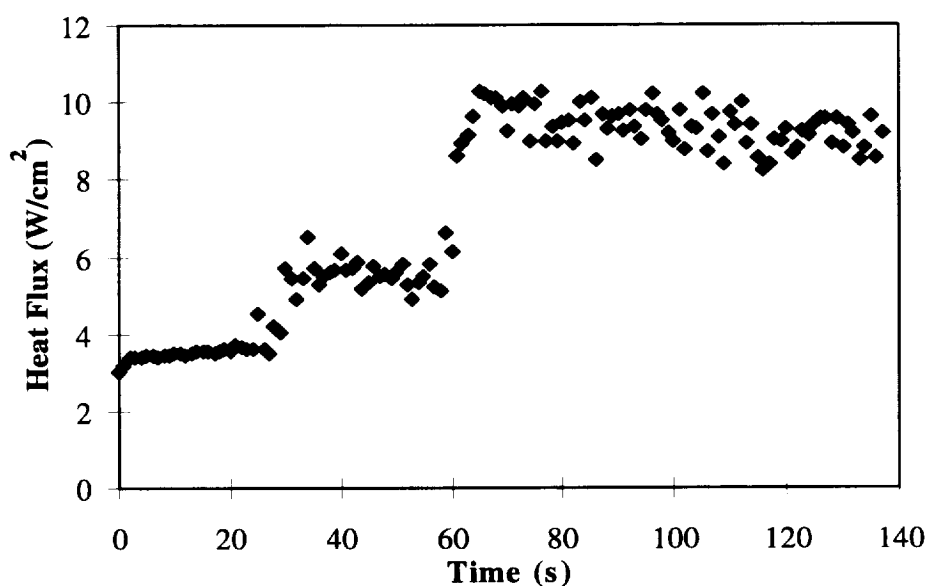


Figure 5.1: Changes in array averaged heat flux over time at  $\Delta T_{\text{sat}} = 17.5$  °C.

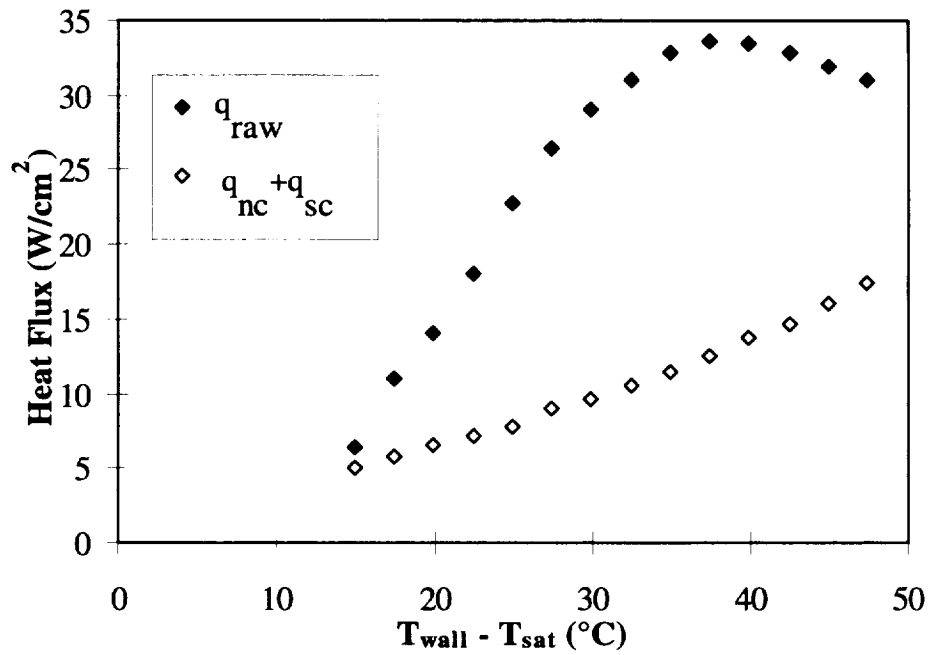


Figure 5.2: Uncorrected boiling curve and substrate conduction data.

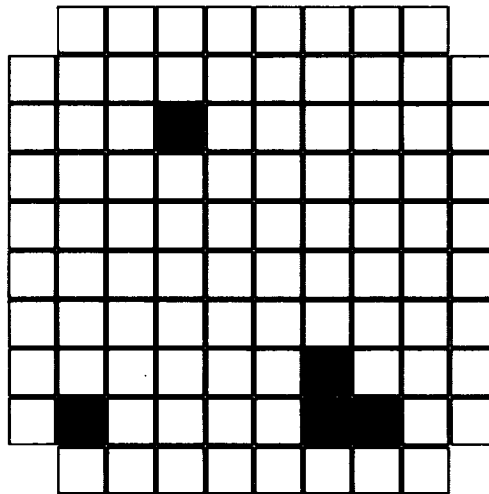


Figure 5.3: Arrangement of 96 heaters in the array, with non-functional heaters represented as black squares

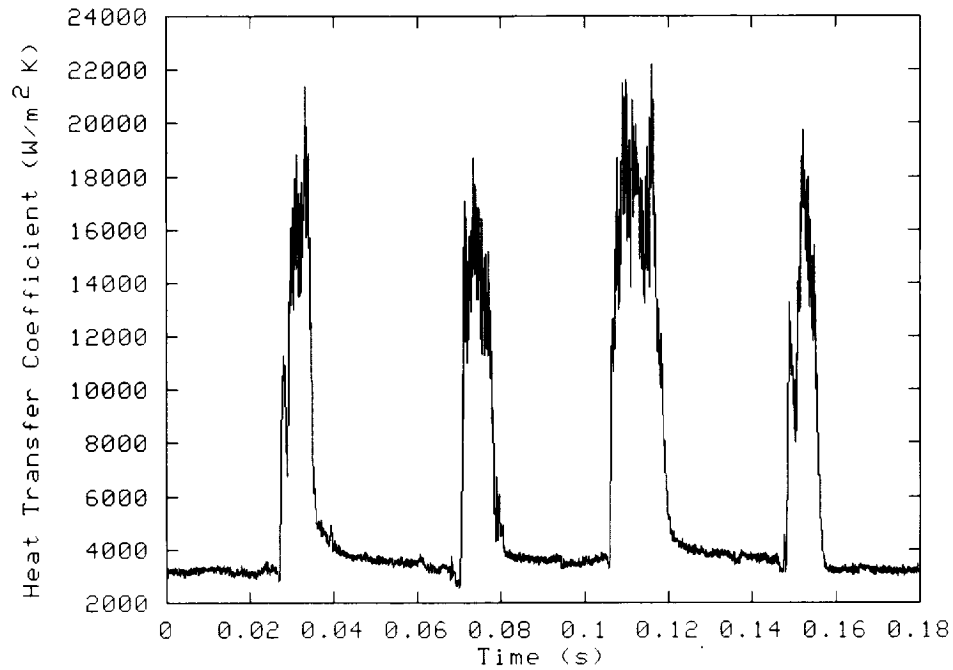


Figure 5.4: Variation in heat transfer coefficient vs. time for a single heater

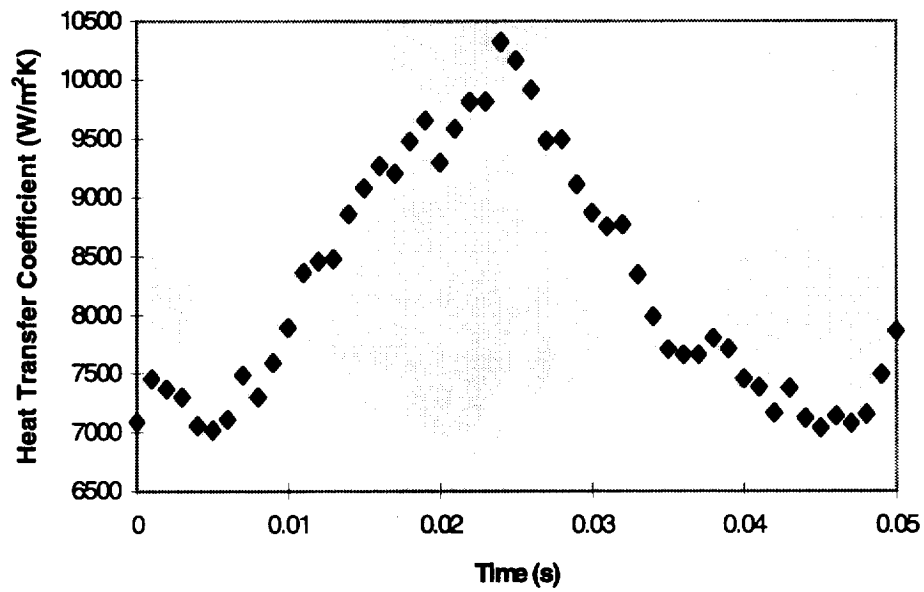


Figure 5.5: Space-averaged heat transfer coefficient vs. Time



Figure 5.6: Surface plots of the local heat transfer coefficient over 64 center heaters (1 major tick mark in the Z-direction =  $5500 \text{ W/m}^2\text{-K}$ , major tick mark closest to horizontal axes =  $0 \text{ W/m}^2\text{-K}$ )



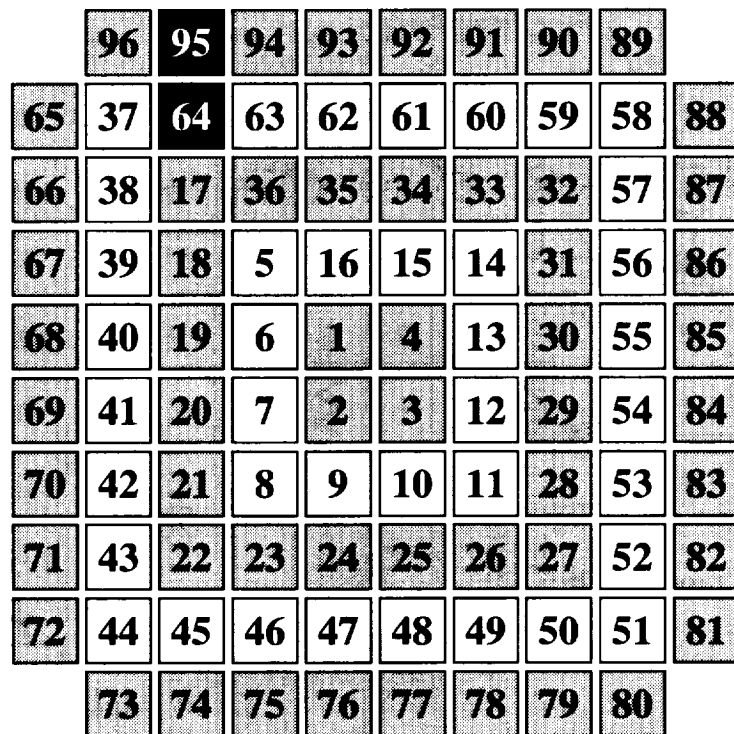


Figure 5.7: Arrangement of 96 heaters in the array, with non-functional heaters represented as black squares.

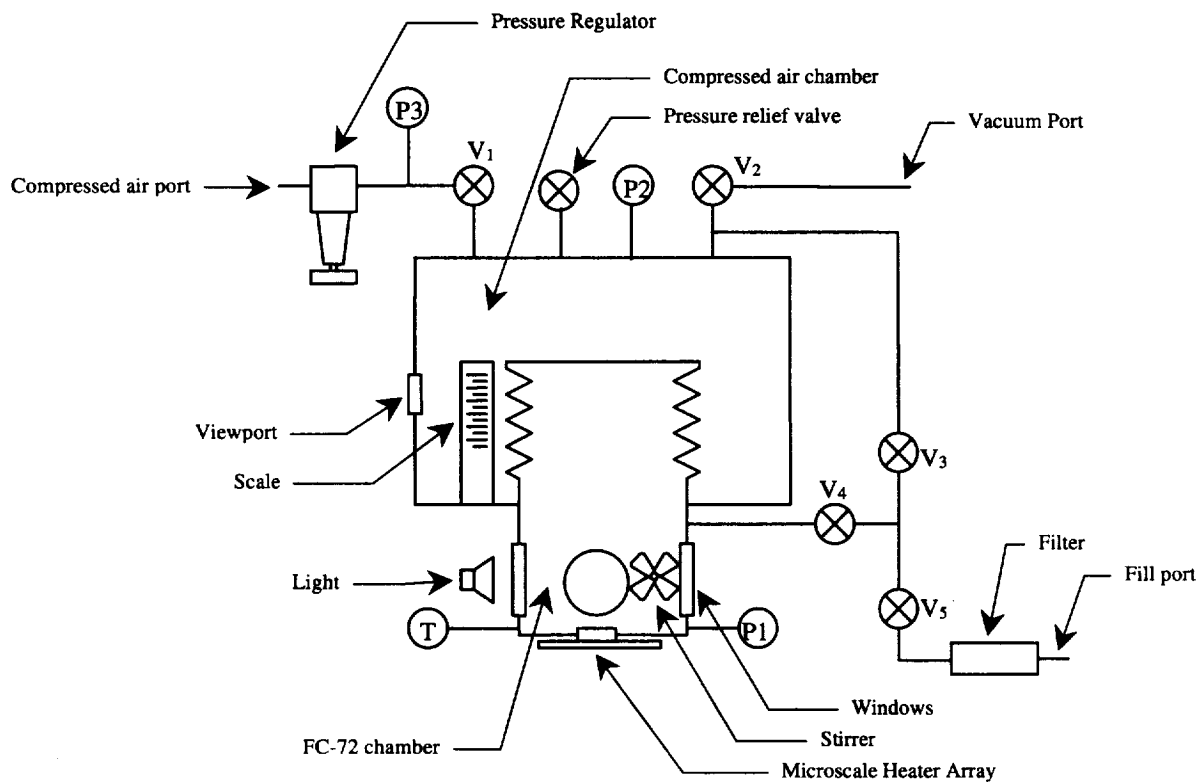


Figure 5.8: Schematic of experimental apparatus.

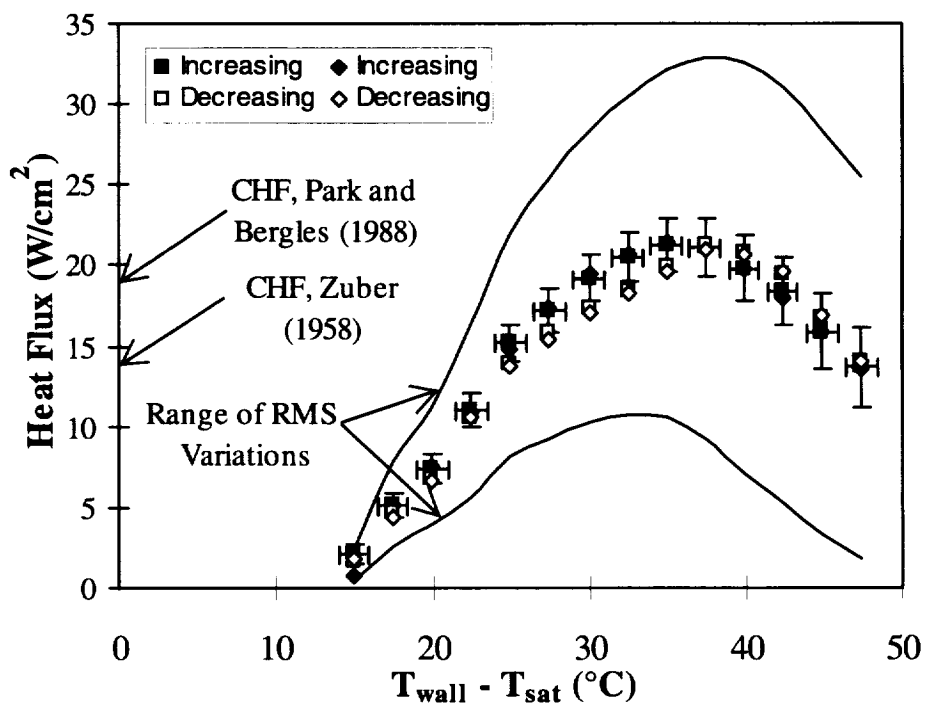


Figure 5.9: Boiling curve showing RMS variation range and uncertainty bars.

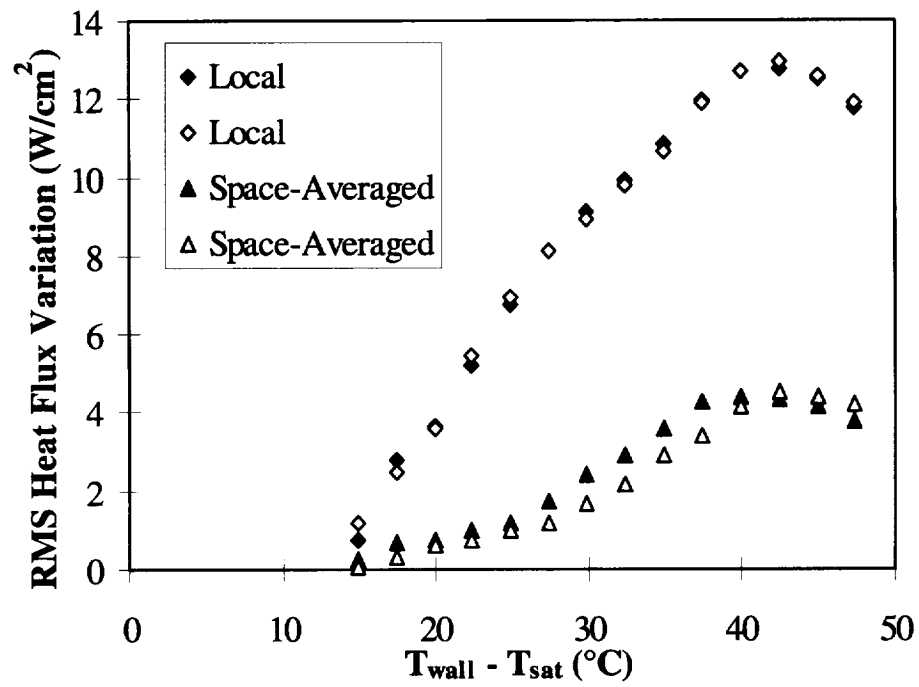


Figure 5.10: Spatially resolved vs. spatially averaged RMS heat flux variation.

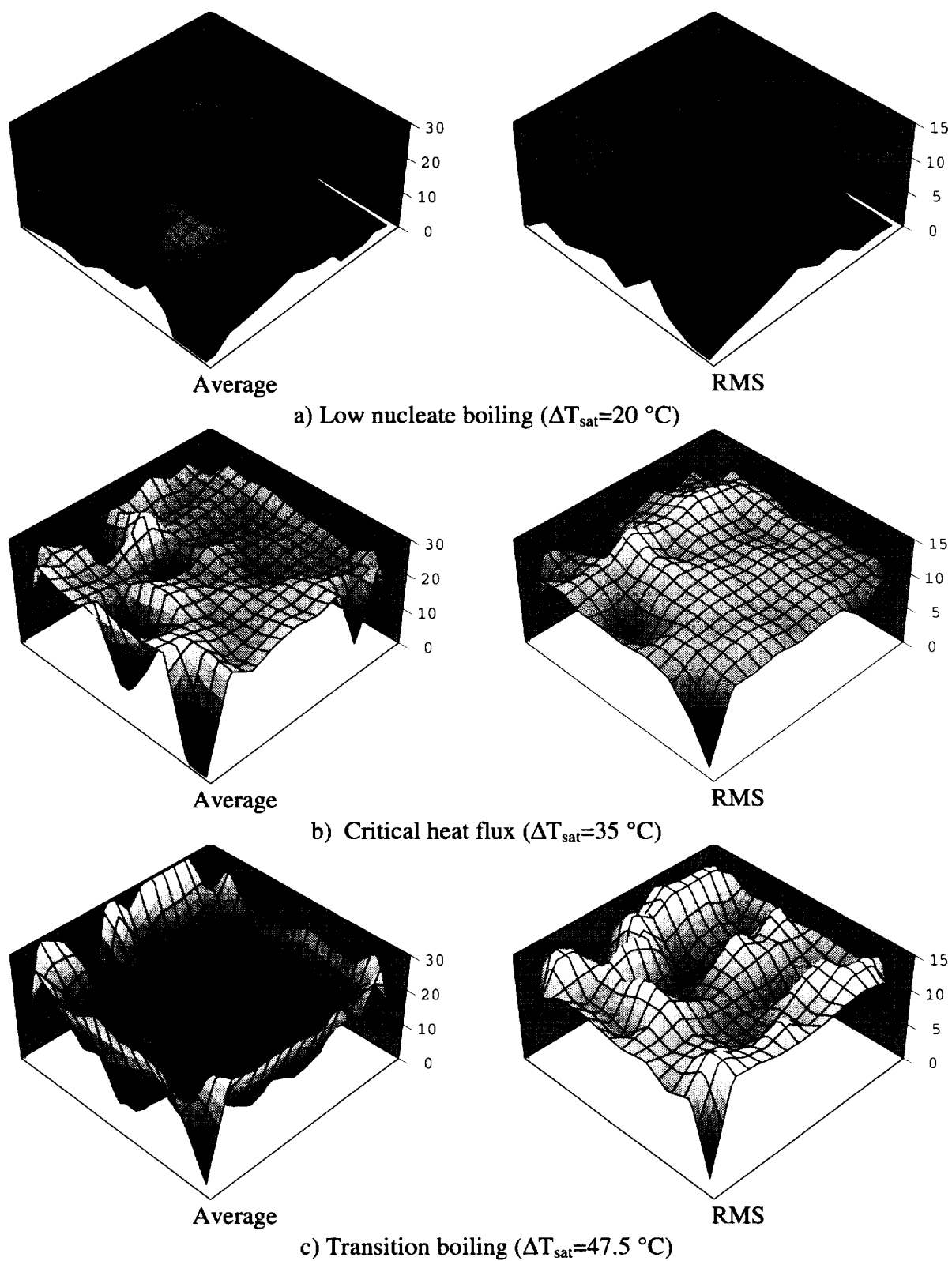


Figure 5.11: Spatially resolved average and RMS variations in heat flux at three superheats

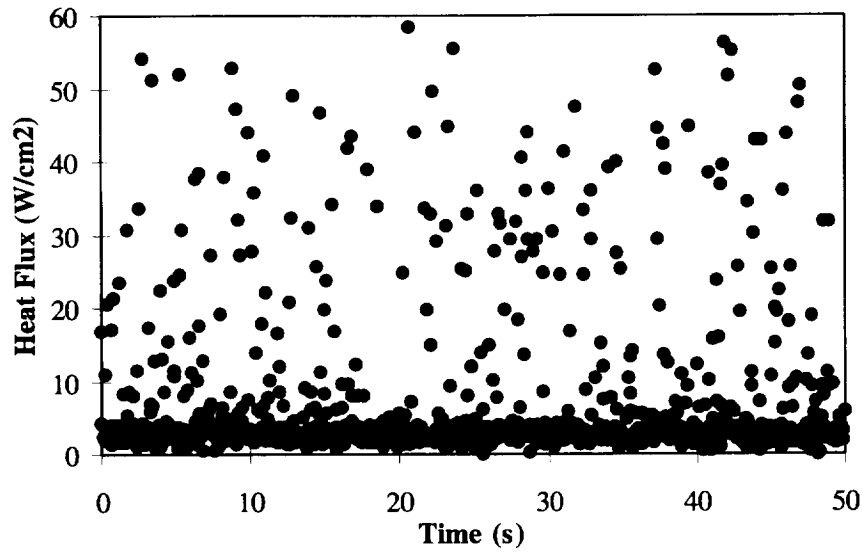


Figure 5.12: Heat flux vs. Time for a heater in the center of the array at  $\Delta T_{\text{sat}}=47.5^\circ\text{C}$  (transition boiling).

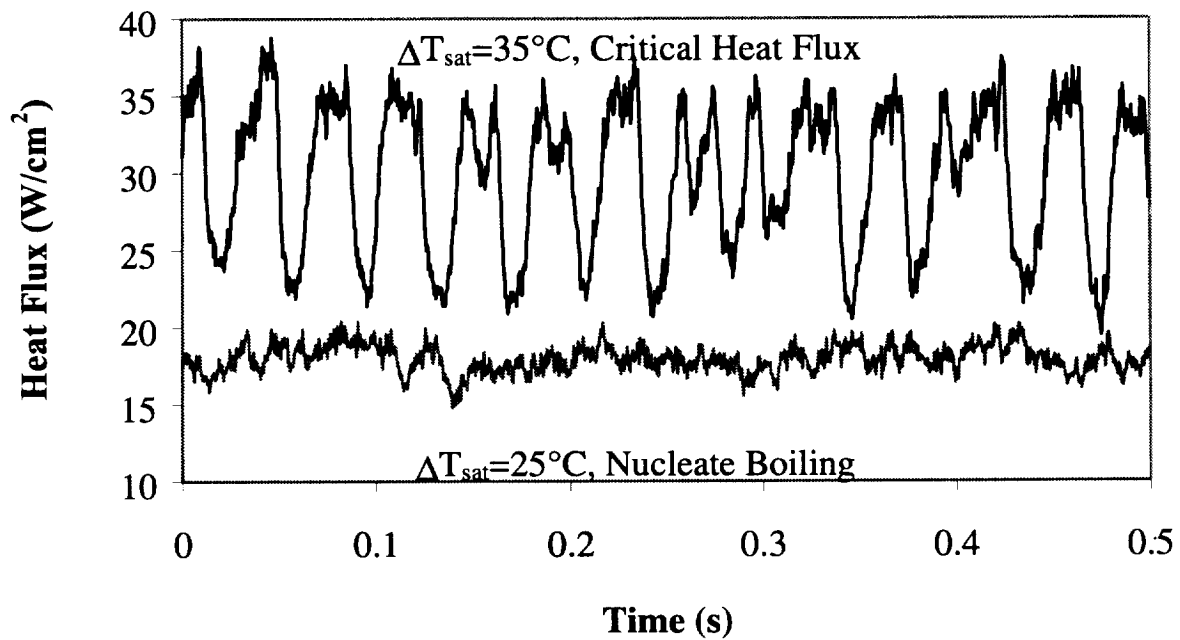


Figure 5.13a: Array averaged, time resolved heat flux vs. time at  $\Delta T_{\text{sat}}=25^\circ\text{C}$  and  $\Delta T_{\text{sat}}=35^\circ\text{C}$ .

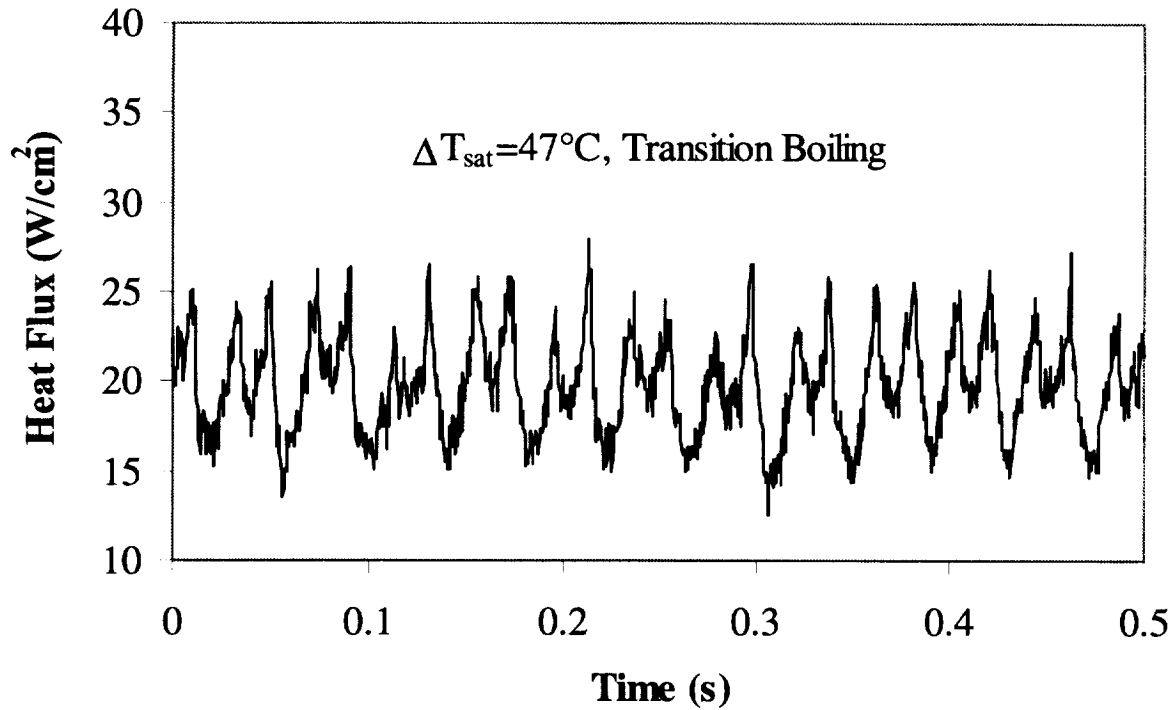


Figure 5.13b: Array averaged, time resolved heat flux vs. time at  $\Delta T_{\text{sat}} = 47^\circ\text{C}$ .

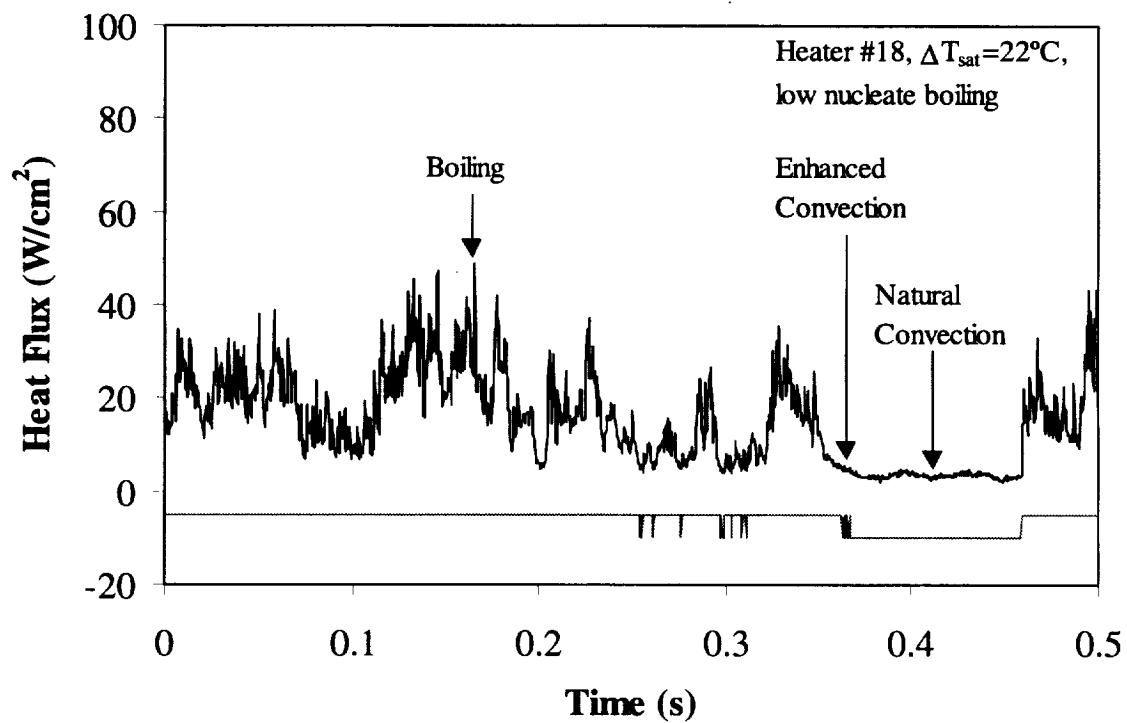


Figure 5.14a: Heat flux vs. time for heater #18 along with the boiling function for  $\Delta T_{\text{sat}} = 22^\circ\text{C}$

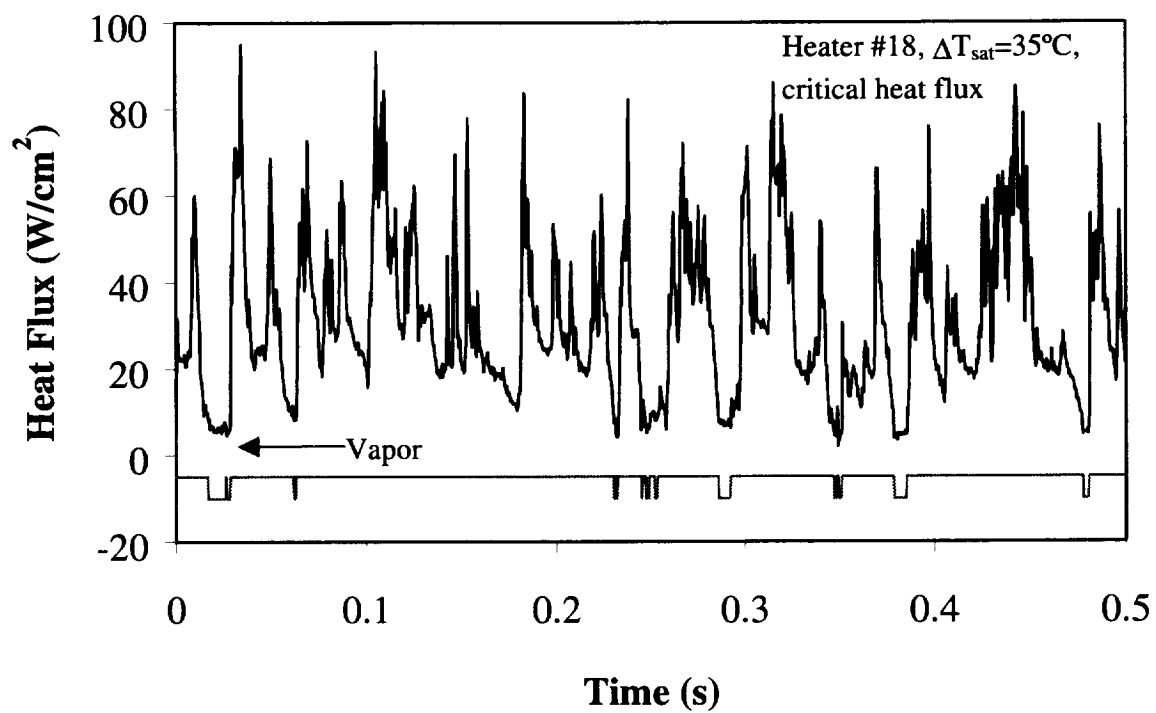


Figure 5.14b: Heat flux vs. time for heater #18 along with the boiling function for  $\Delta T_{\text{sat}}=35^\circ\text{C}$

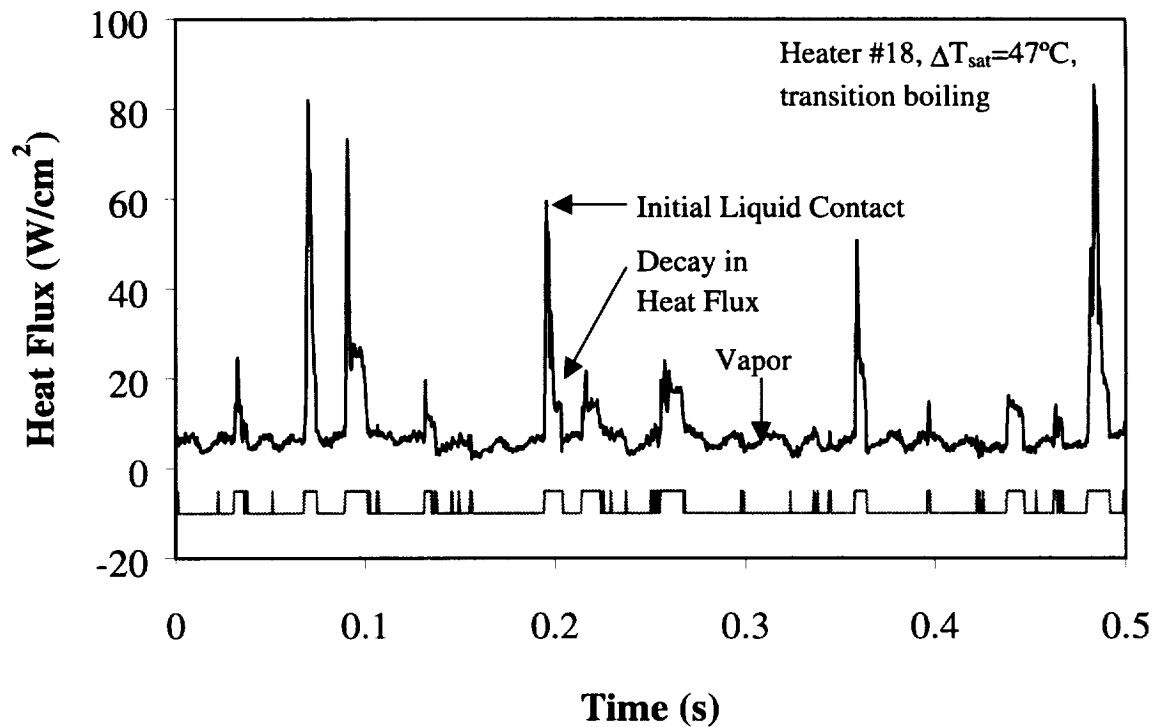


Figure 5.14c: Heat flux vs. time for heater #18 along with the boiling function for  $\Delta T_{\text{sat}}=47^\circ\text{C}$

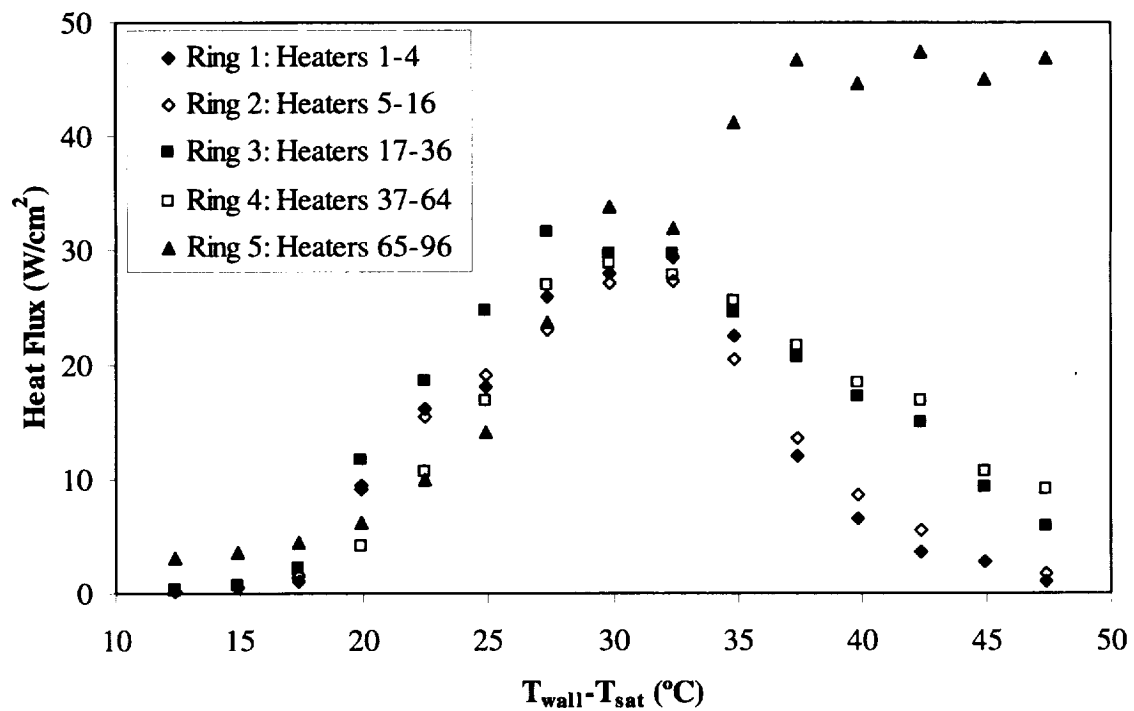




Figure 5.15: Boiling curves for "rings" of heaters

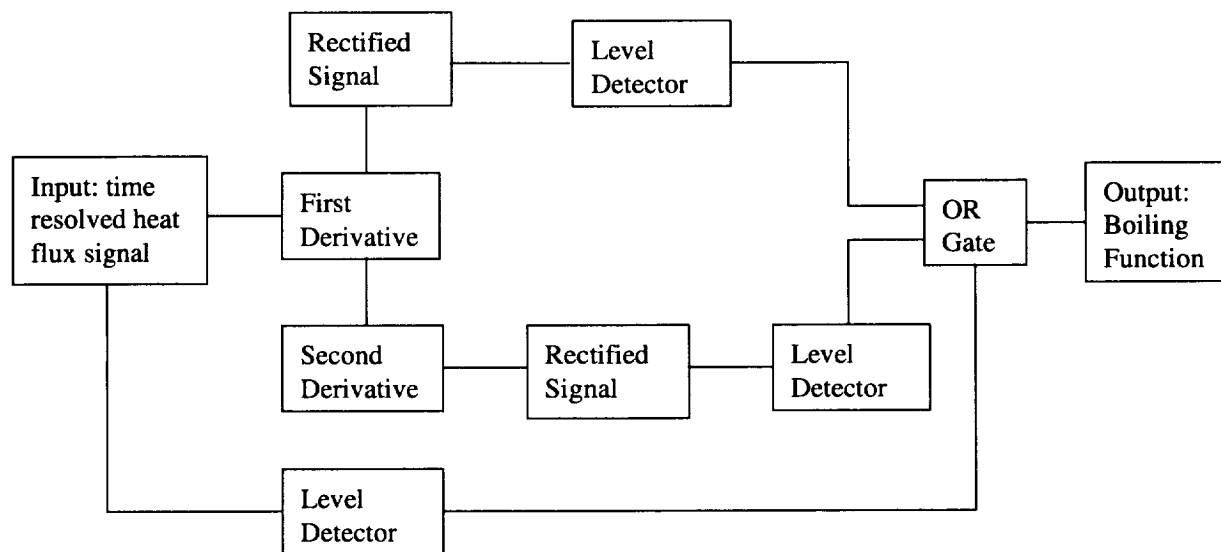


Figure 5.16: Schematic of boiling function generation method

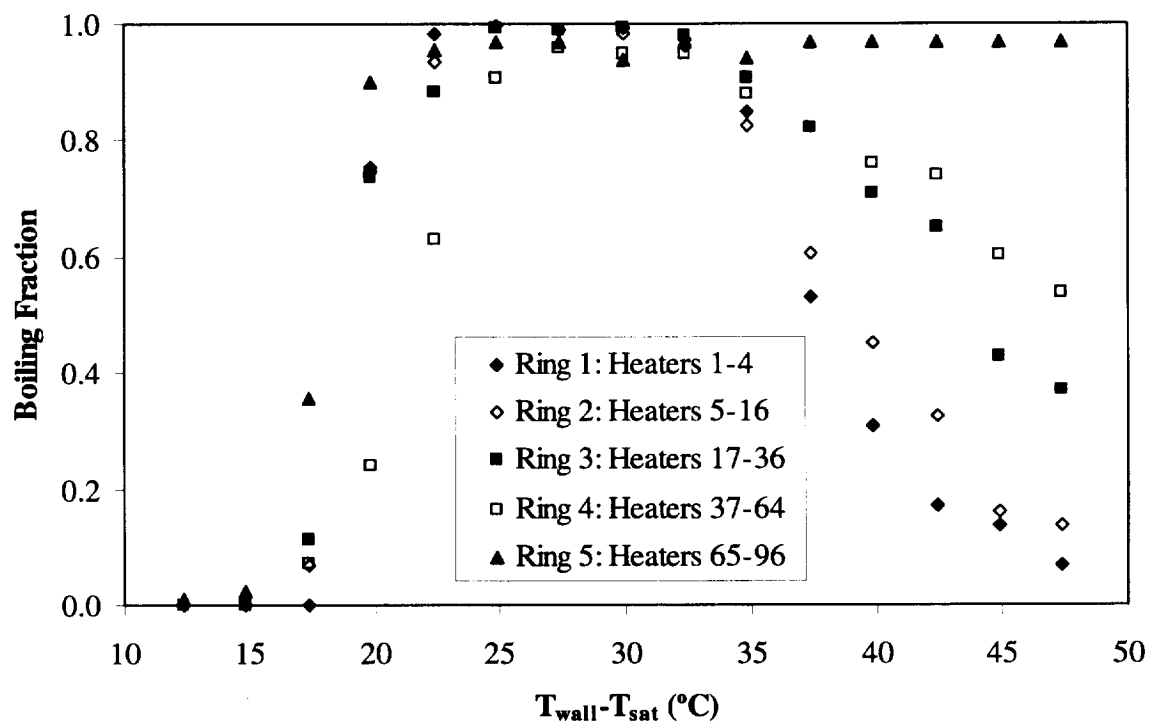


Figure 5.17: Boiling fraction vs. wall superheat for "rings" of heaters.

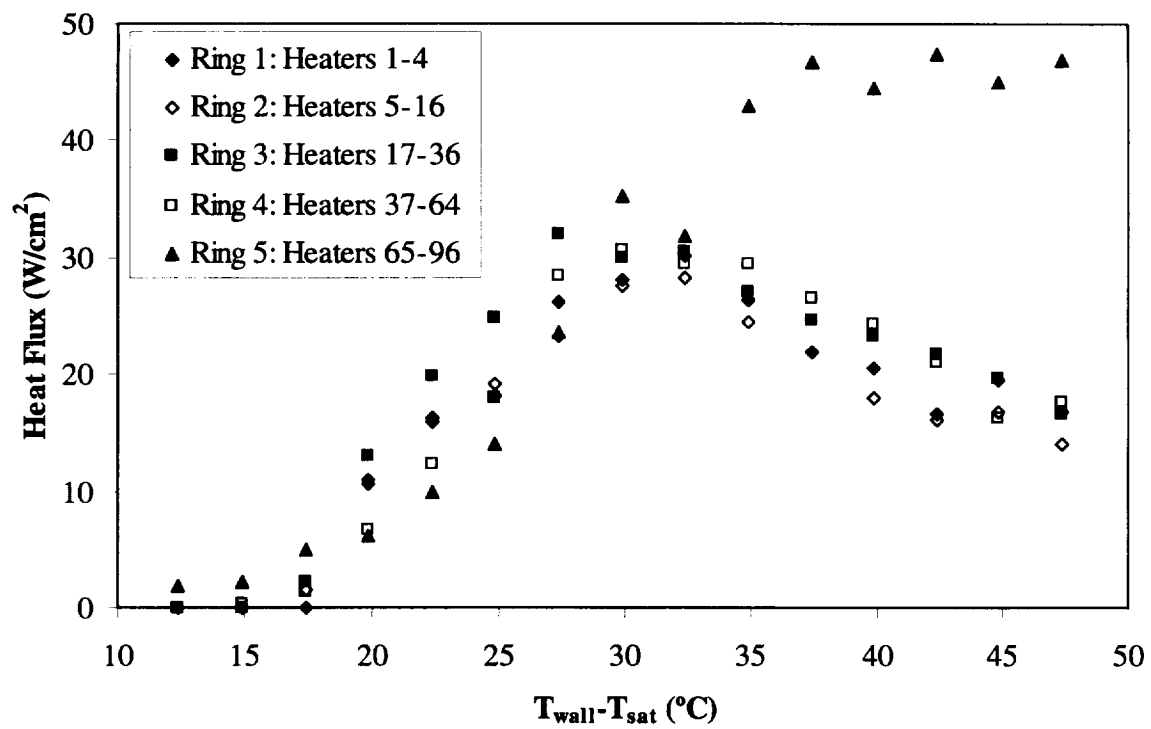


Figure 5.18: Boiling heat flux for "rings" of heaters

## REFERENCES

- 3M Corporation, 1995, *3M Fluorinert Liquids Product and Contact Guide*.
- Alem Rajabi, A.A. and Winterton, R.H.S. (1988a), "Low Thermal Capacity Heaters for Steady State Transition Boiling Measurements", *Experimental Thermal and Fluid Science*, Vol. 1, pp. 405-407.
- Alem Rajabi, A.A. and Winterton, R.H.S. (1988b), "Liquid-Solid Contact in Steady-State Transition Boiling", *International Journal of Heat and Fluid Flow*, Vol. 9, pp. 215-219.
- Chen, J.C. and Hsu, K.K. (1995), "Heat Transfer During Liquid Contact on Superheated Surfaces", *Journal of Heat Transfer*, Vol. 117, pp. 693-697.
- Cooper, M.G. and Lloyd, A.J.P., 1969, "The microlayer in nucleate boiling", *International Journal of Heat and Mass Transfer*, Vol. 12, pp. 895-913.
- Fath, H. S., and Judd, R. L., 1978, "Influence of System Pressure on Microlayer Evaporation Heat Transfer," *ASME Journal of Heat Transfer*, Vol. 100, pp. 49-55.
- Haramura, Y. and Takeno, H., 1997, "Statistical characteristics of parameters affecting pool boiling CHF", *Proceedings of the 1997 Engineering Foundation Conference on Convective Flow and Pool Boiling*, Irsee, Germany.
- Hohl, R., Auracher, H., Blum, J., and Marquardt, W. (1997), "Identification of Liquid-Vapor Fluctuations Between Nucleate and Film Boiling in Natural Convection", *Proceedings of the 1997 Engineering Foundation Conference on Convective Flow and Pool Boiling*, Irsee, Germany.
- Judd, R. L., and Hwang, K. S., 1976, "A Comprehensive Model for Nucleate Pool Boiling Heat Transfer Including Microlayer Evaporation," *ASME Journal of Heat Transfer*, Vol. 98, pp. 623-629.

Kalinin, E.K., Berlin, I.I., and Kostyuk, V.V. (1987), "Transition Boiling Heat Transfer", *Advances in Heat Transfer*, Vol. 18, pp. 241-323.

Kenning, D.B.R., 1992, "Wall Temperature Patterns in Nucleate Boiling", *International Journal of Heat and Mass Transfer*, Vol. 35, pp. 73-86.

Kline, S.J. and McClintock, F.A. (1953) "Describing Uncertainties in Single Sample Experiments", *Mechanical Engineering*, Vol. 75, pp. 3-8.

Koffman, D., and Plesset, M. S., 1983, "Experimental Observations of the Microlayer in Vapor Bubble Growth on a Heater Solid," *ASME Journal of Heat Transfer*, Vol. 105, pp. 625-632.

Lee, L.Y.W., Chen, J.C., and Nelson, R.A. (1985), Liquid-Solid Contact Measurements Using a Surface Thermocouple Temperature Probe in Atmospheric Pool Boiling Water", *International Journal of Heat and Mass Transfer*, Vol. 28, pp. 1415-1423.

Lee, R. C., and Nydahl, J. E., 1989, "Numerical Calculation of Buuble Growth in Nucleate Boiling from Inception Through Departure," *ASME Journal of Heat Transfer*, Vol. 111, pp. 474-479.

Lloyd, J.R. and Moran, W.R. (1974), "Natural Convection Adjacent to Horizontal Surfaces of Various Platforms", ASME Paper 74-WA/HT-66.

Marquadt, W. and Auracher, H. (1990), "An Observer-Based Solution of Inverse Heat Conduction Problems", *International Journal of Heat and Mass Transfer*, Vo. 33, pp. 1545-1562.

Nishio, S., Gotoh, T., and Nagai, N. (1997), "Observation of Boiling Structures", Engineering Foundation Convergence on Convective Flow and Pool Boiling, Irsee, Germany.

Park, K.A. and Bergles, A.E., 1988, "Effects of size of simulated microelectronic chips on boiling and critical heat flux", *Journal of Heat Transfer*, Vol. 110, pp. 728-734.

Rajab, I. and Winterton, R.H.S., 1990, "The two transition boiling curves and solid-liquid contact on a horizontal surface", *International Journal of Heat and Fluid Flow*, Vol. 11, No. 2, pp. 149-153.

Rule, T.D., Kim, J., Quine, R.W., Kalkur, T.S., and Chung, J.N., 1997, "Measurements of Spatially and Temporally Resolved Heat Transfer Coefficients in Subcooled Pool Boiling", Proceedings of the 1997 Engineering Foundation Conference on Convective Flow and Pool Boiling, Irsee, Germany.

Shoukri, M., and Judd, R. L., 1975, "Nucleation Site Activation in Saturated Boiling," *ASME Journal of Heat Transfer*, Vol. 97, pp. 93-98.

Unal, C. and Pasamehmetoglu, K.O., 1994, "Spatial and temporal variation of the surface temperature and heat flux for saturated pool nucleate boiling at lower heat fluxes", HTD-Vol. 273, *Fundamental of Phase Change: Boiling and Condensation*, ASME, pp. 49-56.

Ungar, E.K. and Eichhorn, R., 1996, "Transition boiling curves in saturated pool boiling from horizontal cylinders", *Journal of Heat Transfer*, Vol. 118, pp. 654-661.

Voutsinos, C. M., and Judd, R. L., 1975, "Laser Interferometric Investigation of the Microlayer Evaporation Phenomenon," *ASME Journal of Heat Transfer*, Vol. 97, pp. 88-92.

Watwe, A.A. and Hollingsworth, D.K., 1994, "Liquid Crystal Images of Surface Temperature During Incipient Pool Boiling", *Experimental Thermal and Fluid Science*, Vol. 9, pp. 22-33.

Zuber, N., 1958, "On the stability of boiling heat transfer", *Transactions of the ASME*, Vol. 80, pp. 711-720.

## APPENDIX A: DERIVATION OF OFFSET ERROR

A certain error in the equivalent resistance results from offset voltages in the multiplier chip and the op-amp. It is shown here that these errors can be minimized throughout the operating range of the circuit by carefully adjusting the trimming potentiometers in the circuit. First, an expression for equivalent resistance is derived, and then the effects of offset voltage and trimming adjustments on this expression are evaluated. The multiplier circuit is illustrated in Figure A.1, and can be referred to in the following analysis.

### A.1. DERIVATION OF EQUIVALENT RESISTANCE

Following is a derivation of the multiplier Equivalent Resistance, not accounting for multiplier offsets.

Resistance in terms of voltage from top of circuit to ground, and current through circuit, is

$$R_{eq} = \frac{V_1}{i}$$

Equation A.1

The output voltage response in terms of input voltages for the analog multiplier is

$$W = \frac{(X_1 - X_2) \cdot (Y_1 - Y_2)}{10 \cdot V} + Z$$

Equation A.2

The following inputs will be grounded, so these voltage inputs will be zero.

$$X_1 = 0$$

$$Y_1 = 0$$

$$Z = 0$$

The voltage across the circuit is measured from  $X_2$  to GND, so

$$V_1 = X_2$$

Substitution yields

$$W = \frac{-V_1 \cdot Y_2}{10 \cdot V}$$

Equation A.3

The current through the resistor R is calculated.

$$i = \frac{V_1 - W}{R}$$

Equation A.4

Substituting the expression for W yields

$$i = \frac{V_1 + \frac{V_1 \cdot Y_2}{10 \cdot V}}{R}$$

Equation A.5

Factoring this expression yields

$$i = \frac{V_1}{R} \cdot \left( 1 + \frac{Y_2}{10 \cdot V} \right)$$

Equation A.6

Referring to Equation A.1, and substituting the expression for i yields

$$R_{eq} = \frac{V_1}{\frac{V_1}{R} \cdot \left( 1 + \frac{Y_2}{10 \cdot V} \right)}$$

Equation A.7

Simplifying this expression yields

$$R_{eq} = \frac{R}{1 + \frac{Y_2}{10 \cdot V}}$$

Equation A.8

which is a valid expression for  $R_{eq}$ .

## A.2. DERIVATION OF OFFSET ERRORS

If  $R_{eq}$  changes, the resulting change in temperature will be

$$T_2 - T_1 = \frac{2 \cdot \frac{\Omega}{\text{degC}}}{5 \cdot \frac{\Omega}{\Omega}} \cdot (R_{eq2} - R_{eq1})$$

Equation A.9

assuming that the heater changes by  $2 \Omega/^\circ\text{C}$ , and the bridge ratio is 5:1.

The input offset will have the same effect as a voltage applied to the input  $X_1$ ,

$$X_1 = \Delta X$$

and the output offset voltage will have the same effect as a voltage applied to the input  $Z$ , so that the expression for the multiplier output voltage becomes

$$W = \frac{(V_1 - \Delta X) \cdot Y_2}{10 \cdot V} + \Delta W$$

Equation A.10

Substituting Equation A.10 into Equation A.4 yields

$$i = \frac{V_1 + \frac{(V_1 - \Delta X) \cdot Y_2}{10 \cdot V} + \Delta W}{R}$$

Equation A.11

If this is substituted into Equation A.1, it yields

$$R_{eq} = \frac{10 \cdot V \cdot V_1 \cdot R}{10 \cdot V \cdot V_1 + (V_1 - \Delta X) \cdot Y_2 - 10 \cdot V \cdot \Delta W}$$

Equation A.12

Thus, it can be seen that when offset voltages are present, the equivalent resistance will change when the voltage across the circuit changes. This is not ideal behavior.

### A.3. EFFECT OF TRIMMING ON OFFSET ERRORS

In the experiment, a bias current to the circuit at the top of the equivalent resistance circuit was used to eliminate the multiplier input offset, and the op-amp offset voltage was adjusted in order to null out the multiplier output offset. The effects of these adjustments are calculated here.

The circuit was designed so that the bias current into the circuit,  $I_{OS}$ , would be proportional to the  $Y_2$  input on the op-amp. The proportional factor here is  $i_0$ .

$$i_{OS} = i_0 \cdot Y_2$$



Equation A.13

The current that is going through the multiplier circuit is thus the current going through that side of the bridge,  $i$ , plus the offset current,  $I_{OS}$ . Thus, Equation A.4 becomes

$$i + i_{os} = \frac{V_1 - W}{R}$$

or,

$$i = \frac{V_1 - W}{R} - i_{os}$$

Equation A.14

Substituting Equation A.10 into Equation A.14 yields

$$i = \frac{V_1 + \frac{(V_1 - \Delta X) \cdot Y_2}{10 \cdot V} - \Delta W}{R} - i_{os}$$

Equation A.15

The equivalent resistance that the op-amp sees will be

$$R_{opamp} = \frac{V_1 + V_{os}}{i}$$

Equation A.16

where  $V_{OS}$  is the offset voltage of the op amp.  $V_{OS}$  is adjustable using an offset-null circuit attached to the op-amp. The resistance  $R_{opamp}$  will determine how the op-amp responds to changes in the system. Substituting Equation A.15 into Equation A.16 and expanding yields

$$R_{opamp} = \frac{10 \cdot V \cdot (V_1 + V_{os}) \cdot R}{10 \cdot V \cdot V_1 + (V_1 - \Delta X - 10 \cdot V \cdot i_o) \cdot Y_2 - 10 \cdot V \cdot \Delta W}$$

Equation A.17

Offset effects will disappear if the following conditions are true.

$$V_{os} = \Delta W$$

$$\Delta X + 10 \cdot V \cdot i_o = \Delta W$$

Equation A.17 reduces to Equation A.8 when these conditions are true.

The circuit that is used to supply this offset current does not actually supply a constant current. This current depends on the voltage difference between the control voltage and the

voltage at the top of the  $R_{eq}$  circuit. This voltage can either increase or decrease with increasing control voltage. The following equation describes the behavior of the offset voltage.

$$i_{os} = \frac{Y_2 \cdot C \cdot V_1}{R_c}$$

Where  $Y_2$  is the control voltage,  $C$  is a constant that can vary between 1 or -1 using a potentiometer as a voltage divider, and  $R_c$  is the resistance between the control voltage and the top of the  $R_{eq}$  circuit.

Substituting into Equation A.17 yields

$$R_{opamp} = \frac{10 \cdot V \cdot (V_1 + V_{os}) \cdot R}{10 \cdot V \cdot (V_1 - \Delta W) + (V_1 - \Delta X) \cdot Y_2 - 10 \cdot V \cdot R \cdot \frac{Y_2 \cdot C \cdot V_1}{R_c}}$$

We want this to reduce to the equation for ideal op-amp performance, Equation A.8 or an equivalent form. This will occur when the following conditions are met.

$$V_{os} = \Delta W$$

Condition 1

$$V_{os} \cdot Y_2 = 10 \cdot V \cdot R \cdot \frac{Y_2 \cdot C \cdot V_1}{R_c} - \Delta X \cdot Y_2$$

Condition 2

and

$$Y_2 \cdot C \text{ is much greater than } V_1 \text{ and}$$

Condition 3

$$\frac{10 \cdot V \cdot R \cdot C}{R_c} = \Delta X$$

Condition 4

Condition 3 can be met by using a very large resistor, so that  $Y_2 C$  will be much larger than  $V_1$  at the low voltages where offset error is significant. The op-amp offset voltage and the potentiometer that controls the value of  $C$  can be carefully adjusted until the other two values are met. Condition 4 does not seem to cause significant error in the circuit under experimental conditions.

This analysis will break down at small values of  $C$ , because condition three will lose validity.

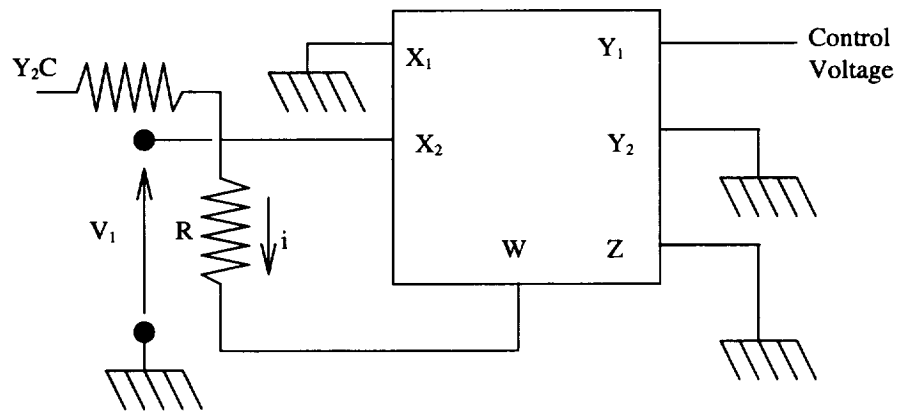


Figure A.1: Equivalent resistance circuit using multiplier chip

## **APPENDIX B: SOFTWARE OPERATING INSTRUCTIONS**

### **B.1. CAL**

After loading the CAL program, the window shown in Figure B.1 will appear. The first step is to go to the setup window by clicking on "Setup." The setup window is shown in Figure B.2. After making changes to the setup, it will be convenient to save those changes in a setup file so that they need not be entered again. This will be discussed at the end of the section.

Typically, the "Range of heaters" option will be used. The "All Heaters" option selects a range of heaters that is based on a variable in the program that may have been set when a different heater array design, with a different number of heaters, was in use. In other words, this option is obsolete, and may be eliminated in future revisions of the program.

The heater numbering system within all the programs starts at zero. For example, if the program is being used with a heater array that has 96 heaters, heater "0" will be entered as the first heater, and heater "95" will be entered as the last heater.

The "Specific Heaters" option is only used when it is desired to calibrate a specific group of heaters. Heaters are added to the list of heaters to calibrate by putting the heater number in the text-box and pressing "Add". The heater number will be added to the list below. Heaters in the list can be selected with the mouse, and then the "Delete" button can be used to remove them from the list.

During the calibration, the program will incrementally increase or decrease the command voltage to a heater control circuit until the heater voltage reaches the threshold voltage that is shown in the setup window. As described in the body of the thesis, the threshold voltage is an

important parameter in the calibration process. Presently a threshold voltage of 900 mV is typically used.

If it is desired to perform a rough calibration without turning on the calibration bath in order to check the function of the equipment, one can set the threshold very high so that temperature of the heater will increase significantly during the calibration of each heater. This temperature increase tends to be somewhat consistent from one heater to the next, so that all the heaters will end up calibrating to nearly a constant temperature. This can be used at least to make sure all the heaters are operating and that the control voltages are roughly the same.

The Maximum DeltaV value should be around 5 to 10 volts. Since the program uses the bisection method to find the value of  $V_{cmd}$  that results in the threshold voltage, cutting this value in half will add only one additional bisection step. The size of the deltaV value needs to be minimized because it contributes slightly to the uncertainty in temperature.

The voltage step rate is a clock-controlled rate that limits the speed that the program can step from one  $V_{cmd}$  value to the next as it performs the bisection method. It has been found that a better method of limiting the step rate is to adjust the hardware scan rate and the scans per data point. Twenty scans per data point at a rate of 80 scans per second is a reasonable value. This will result in a voltage step rate of about 4 steps per second. It is better to use a low scan rate because large transients occur in the circuit output at the time when the control voltage changes. If the scan rate is too fast, then with only 20 scans, all the scans might occur during the transient, which would result in an erroneous reading of the heater output voltage. Enough time must be given for the transient to settle, and the majority of the data must be acquired after the transient has settled.

In order to find the best scan rate, an oscilloscope can be connected to the heater output, and the output watched to observe the length of the transients compared to the steady portion. The final output voltage before the computer moves onto the next point can be observed to see that it is very close to the programmed threshold voltage.

The ADC offset needs to be set to 16 when using the daqbook with the expansion chassis, because the first channel of the expansion chassis is addressed as channel 16. Otherwise, the calibration will fail.

The calibration settings can be saved so they can be quickly recalled and modified later. One convention for naming calibration files is to save them using the date that they were created as a file name. Thus they can be more easily correlated with log book entries and with data files that are subsequently created using the earlier calibration files.

Note that the Start button on the Heater Array Calibration window will not become active until the OK button on the setup menu is pressed.

Before starting a calibration, the temperature bath should be allowed to reach a steady temperature, and the temperature should be entered into the Temperature box on the main window. A valid file name must be entered before the calibration can begin. Any comments about special calibration conditions can be entered in the comments box.

When all preparations have been made, press the Start button. If any errors occur during the calibration of a specific heater, a message will be displayed in the error box, e.g. "Calibration failed, heater #1."

While the heater array is calibrating, the heater number that is currently being calibrated will be displayed in the main window. Also, the previously calibrated heater and the resulting  $V_{cmd}$  value will be calibrated.

### **B.1.1. Description of Controls in "Heater Array Calibration" Window**

The "Heater Array Calibration" window, shown in Figure B.1, contains the following controls, which are described below.

**Temperature.** Used to enter the temperature of the present calibration point. This temperature is saved with the calibration file and used by the CONTROL program to interpolate the control voltages between calibration file temperatures.

**Status.** Indicates whether the program is "Calibrating" or "Inactive".

**Test Time.** Displays the number of seconds that have passed since the calibration for a given temperature began.

**Calibrating Heater.** Displays the number of the heater that is presently being calibrated. This number counts the first heater as heater number zero.

**Vcmd + Repetition #.** These boxes are now meaningless, due to program revisions.

**Final Calibration for Previous Point.** The text boxes that follow this label contain information about the last heater that was calibrated.

**Heater #.** Displays the number of the heater that was last calibrated

**Average Vcmd.** Displays the  $V_{cmd}$  value that resulted from the previously calibrated heater. This is the value that will be saved in the calibration file.

**File Name.** Used to enter the file name of the calibration file to save. These files generally end with ".CAL".

**Comments.** Used to enter any additional information that is desired to be saved with the calibration file, such as special experimental conditions.

**Errors.** Lists heaters which did not calibrate successfully. Most recent errors are added to the beginning of the list.

**Setup.** Opens the "Setup" window, which allows the user to configure calibration parameters.

**Start.** Start calibrating the heater, saving calibration data when it is finished.

**Stop.** Halt the calibration

**Exit.** Exit the program

### **B.1.2. Description of Controls in "Heater Calibration Setup" Window**

This window, shown in Figure B.2, allows the parameters of the calibration routine to be adjusted. It contains the following controls.

**Active Heaters.** This label provides three options concerning which heaters will be calibrated: All Heaters, Range of Heaters, and Specific Heaters

**All Heaters.** Calibrates ALL heaters, from heater zero up to the maximum number of heaters specified in the program. Since the number of heaters in the heater array has changed, and the number specified in the software may not have changed, this may not be a reliable option.

**Range of Heaters.** Allows the user to specify the range of heaters which will be calibrated by entering the first and last heater in the range. Heater numbers are counted starting at zero, not one.

**Specific Heaters.** Allows the user to add specific heaters to a list of heaters to be calibrated. Heater numbers are entered in the small box, and the "Add" button is used to add that heater to the list, which is displayed in the large box below. Heater numbers in the box can be highlighted and removed using the "Delete" button.

**Threshold Voltage.** The heater voltage at which the Vcmd values are saved is entered in this box.



**Maximum DeltaV.** The size of interval where the bisection method is satisfied is entered in this box.

**Voltage Step Rate.** The rate that the calibration program advances to the next  $V_{cmd}$  value in the bisection method. This must be fast enough that calibration is accomplished in a reasonable amount of time, but slow enough that the output voltage has time to settle at each  $V_{cmd}$  value.

**Scans / Data point.** Number of output voltage values to digitize at each  $V_{cmd}$  value. The scans are averaged at each datapoint and used to determine what the output voltage is.

**Hardware Scan Rate.** The number of scans per second at which these voltage values are digitized.

**ADC offset.** This value must be entered when using the DAQbook data acquisition system with the expansion cards. A value of 16 is used if the expansion cards are being used. Otherwise, zero is used. This number arises from the fact that in the functions that address the expansion cards, the first channel on the first expansion card is addressed as channel 16, not channel zero, so the A/D addresses must be offset in the software to account for that.

**OK.** The current settings are saved and the setup window is closed when this button is pressed.

**Cancel.** The previous settings are restored and the setup window is closed when this button is pressed.

**Load Setup.** The user is prompted for the name of a previously saved setup file, with the extension ".CAL", which contains the setup data given in this window when this button is pressed.

**Save Setup.** The user is prompted for a file name which the current settings will be saved under.

**Restore Defaults.** This button restores the values that were present when the program was started.

## **B.2. CONTROL**

The CONTROL program is used to set the heater array to the temperatures for which it was calibrated using the CAL program, to acquire heat transfer data while the heater array is operating, and to perform various data reduction tasks.

After the program is loaded and the window shown in Figure B.3 appears, the first step is to set up CONTROL for the hardware that is being used. Press the Setup button to display the setup window, which is shown in Figure B.4. After setting all the options, the settings can be saved for recall next time the program is used.

Calibration files can be loaded into a set of calibration tables within the program by selecting a table number and then clicking on the Browse button to select a calibration file. Up to 16 calibration files can be loaded in this manner. The temperature for each calibration file is stored with the file, and the computer will eventually use that temperature and the  $V_{cmd}$  values within the calibration file to calculate the proper  $V_{cmd}$  values for any temperature that is entered that falls between the highest and lowest calibration file temperatures. The order that the calibration files are listed does not matter. It also does not matter which tables are left blank, whether they are at the beginning, at the end, or in between non-blank tables.

The A/D options include No Hardware, for when the program is being tested without hardware or when no A/D hardware is connected, DAQBook, for when the DAQBook hardware is being used, and Custom A/D, for when the high-speed A/D boards are being used.

The D/A options determine what method is used to control the heater control circuits. Either No Hardware or Computer Control Board can be selected.

There are presently several gain options given in the control setup window, but the only valid option at this time is a gain of 1. Since the heater voltages presently vary over the entire dynamic range of the A/D system, there is no need for higher gains, and no such need is anticipated in the future. Future revisions of the software may eliminate the gain option.

The sampling rate should be set to a value that provides slightly more resolution than is necessary, but does not provide too much data to conveniently process. The maximum sampling rate using the DaqBook with 96 heaters is 1000 samples/s. However, the CONTROL program is not able to attain this sampling rate because it cannot download data from the DaqBook buffer quickly enough.

A program called DAQVIEW can be used to collect data at the maximum speed of 1000 samples/s, if it is desired. The binary files produced by this program are compatible with all the post-processing procedures of the CONTROL program, as long as the appropriate .TAG file is included. This can be done by setting up CONTROL to sample all 96 heaters at 1000 samples per second and starting the data acquisition procedure. When the procedure fails due to a buffer overflow, the binary file that CONTROL produces will be useless, but the .TAG file that is generated can be used with a binary file from DAQVIEW. This can only be done if this file was generated by sampling the same number of heaters and the sampling time and sampling rates were the same as what was specified in CONTROL when the .TAG file was generated.

The Custom A/D board is capable of much higher sampling rates. When using the CONTROL program, the custom A/D board always samples at 10,000 samples/s. Within control, however, sampling rates of 1250, 2500, 5000, or 10,000 may be selected. These

sampling rates are achieved by physically sampling the data at the highest rate of 10,000 samples/s, but then only downloading every 2nd, 4th, or 8th point from the buffer. This merely reduces the size of the data file, and insures that the time skew will be minimized.

The DAQBook does not have a minimum sampling rate. However, the custom A/D system has a minimum sampling rate of 1250 samples/second.

The duration that the DAQBook can continue to collect data is limited only by the amount of disk space on the computer, since the DAQVIEW program can stream data from the A/D converter to the hard disk at 100,000 samples/s, or about 1000 samples/s per heater. The custom A/D converter can only continue to collect data for 1.6 seconds when physically sampling at 10,000 samples/s per heater.

The only trigger source that is presently available for starting the A/D system is a manual trigger using the software. An external trigger for synchronizing with other events could be added later.

In the CONTROL program, a channel offset of 16 is necessary when collecting data from the DAQBook with the expansion cards installed. A channel offset of zero is used for the custom A/D system or for the DAQBook base unit.

The All Heaters, Range of Heaters, and Specific Heaters options are similar as for the calibration program, except in this case it determines which heaters data will be collected from. It does NOT determine which heaters will be turned on. The calibration file that is loaded determines this.

The only data file format that is presently available is the binary format, and this option will probable be eliminated in future revisions.

Once the options are set, they can be saved for later recall using the Save Setup button, and then they can be put into effect in the CONTROL program by pressing OK.

The temperature of the heater can be set using two methods. One method is to use the Table Number option. Click on the Table Number option button and enter a table number which corresponds to the tables that were specified in the setup. Once a table number is specified, click on Set Heaters to reprogram the computer control board to output the specified calibration table to the control circuits, and thus set the heaters to the specified temperature.

The other method is to set the temperature. To use this option, click on the Temperature option button and set the temperature to any value between the lowest and highest temperature that is given in the calibration files. When Set Heaters is selected, The program uses an interpolation scheme to interpolate between the nearest temperatures above and below the given setpoint. The interpolated  $V_{cmd}$  values are then programmed into the computer control board and output to the control circuits so that the heater array is set to the specified temperature.

Once the temperature is set, data can be acquired. A file name is entered in the Data File Name text box, and the Start button is pressed. If the daqbook is being used, the user must wait until data acquisition is complete before resetting the temperature. If the custom A/D board is being used, the user can reset the temperature after the status window displays "Downloading Data", which means that the A/D boards have finished collecting data, and the computer is merely downloading the data from the A/D buffers to the computer's memory.

Every time the user clicks on Setup and then on OK, all the hardware is reset, so that the computer control board outputs zero  $V_{cmd}$  values and the heaters are turned off. In order to keep the user from accidentally taking data without first setting the heaters, the user is required to

click on Set Heaters before clicking on the Start button which begins the data acquisition process.

### **B.2.1. "Control" Window Controls**

The "Control" window, which is shown in Figure B.3, contains the following controls.

**Status.** This text box indicates whether data is being collected, the system is waiting for another data point, downloading data from the high-speed A/D, or Inactive.

**Temperature.** When the button to the left of this label is active, then a temperature can be entered into this box and the program will interpolate between the two closest temperature which bracket the entered value. This interpolation should be done between calibration points that are only 5 to 10 degrees apart. The error due to this interpolation is vary small. The interpolated voltage table is then used when the heater temperature is set. In order use this option, at least two calibration files must have been loaded, and the selected temperature must fall somewhere between the calibration temperature of two of the calibration files.

**Table Number.** When the button to the left of this label is active, the temperature is set by selecting one of the calibration files that have been loaded in the Setup window.

**Automation Setup.** This command button will cause the Data Acquisition Automation window to be displayed, so that the user can set up an automatic data acquisition procedure.

**Errors Last Run.** Errors during data acquisition will be displayed in this text box.

**Data Acquisition File Name.** The file name where the data acquisition data will be stored Is entered in this box.

**Comments.** Any comments which need to be attached to the data file can be entered here.

**Set Heaters.** This button causes the heater temperature to be set to whatever is specified in the "Temperature" or "Table" boxes.

**Start.** This button causes a set of data to be captured and stored in the specified file name. This button will be available until the "Set Heaters" button and the "Setup" button have been pressed.

**Stop.** This button stops data acquisition before it is complete.

**Setup.** This button causes the setup window to be displayed, so that various parameters can be adjusted.

**Convert.** This button causes the "Convert Binary to Text" window to be displayed, so that the user can perform various data reduction operations on the binary files.

**Exit.** This button terminates the program.

#### **B.2.2. "Data Acquisition Setup" Window**

This window, shown in Figure B.4, allows the user to configure the data acquisition software for the data acquisition hardware that is connected to the computer, and for the number of heaters that are being sampled.

**Calibration Files.** The number box allows one out of 16 calibration tables to be selected. The text box to the right of the number box allows a calibration filename to be selected for the selected calibration table.

**A/D Hardware Type.** Allows the A/D hardware type to be selected. The choices are no hardware, DAQBook, and Custom A/D board.

**D/A Hardware Type.** Allows the D/A hardware type to be selected. The choices are no hardware, or the computer control board.

**Gain.** The gain that is used for A/D using the present devices is always one, so this is the only choice given.

**Sampling Rate.** The number of samples per second for each heater.

**Duration.** The length of time over which data is acquired

**Trigger Source.** The source of the signal that will begin data acquisition

**Channel Offset.** A number of 16 is used when the DAQBook is used with its expansion cards, or zero when the DAQBook base unit only is used, or when the custom A/D system is used.

**All Heaters.** This option causes data to be acquired from all the heaters. A more reliable choice is the Range of Heaters option, so that a specific range of heaters can be selected.

**Range of Heaters.** This option allows a range of heater numbers to be selected for data input.

**Specific Heaters.** This option allows any combination of specific heater numbers to be selected for input.

**Data File Format.** The only data file format currently available is a binary file format.

**Comments.** This box is for any extra information that the user wants to attach to the setup file.

**Save Setup.** The user is prompted for a file name where the setup information is saved.

**Load Setup.** The user is prompted for the name of a setup file to load

**Restore Default.** The setup information is restored to the values at the time the program was started.

**OK.** Uses the new settings and returns to the main window

**Cancel.** Discards the changes and goes back to the main window.

**A/D hardware setup.** Options for advanced setup of the A/D hardware

**D/A hardware setup.** Options for advanced setup of the D/A hardware

### **B.3. POST-PROCESSING**

An important feature of the CONTROL program is the ability to perform post-processing operations on the raw data files. To display the post-processing window, press the Post-



Processing command button. The window shown in Figure B.6 is displayed, which allows the user to select a number of post-processing options.

In order to perform an operation in the post-processing window, first set the file format option to the desired format. Then click on the Input File box to select a binary data file to process. The next section discusses each of the operations that can be performed. Select the operation that you desire to perform, and check the next section to see which other text boxes need to be update to insure correct output.

There are four boxes which must always have the correct information in them to obtain proper results. The offset file, resistance coefficient, resistance constant, and area file names need to be supplied to accurately convert the raw voltage data into heat flux. If any of these file names are not entered, errors will occur when the post-processing operation is started.

The following data reduction operations can be performed. These operations can be selected from the list in the "File Formats" box. The following information is given: the name of the format, a description of the format, a list of the other text boxes that are referenced by the operation, and the file extension that is typically used for the format.

Format	Standard
Description	This option outputs tab-separated data that can be imported into a spreadsheet program. The top row consists of heater number labels, and the left column consists of the time. The columns consist of the heater heat flux at various times. This format is useful for spreadsheet manipulation. The units are given in seconds and $W/cm^2$
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	.TRD

Format	Matrices
Description	This file presents the heat flux information as a map of the heat flux for each heater in the array. The data file consists of a series of 10x10 matrices which give the heat flux values for each heater in the array. The file presents a matrix for each time step, starting at zero .
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	.MAT

Format	GNU Plot
Description	
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	.GPD

Format	MPEG Movie
Description	This file saves the data in a format which can be read by GNU plot. It also saves a script which can be used by GNU Plot to plot each time step and save it to disk as a separate file, and DOS batch file which crops the image files and combines them into an MPEG movie using some free software utilities.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	

Format	Image Tool
Description	This file can be used by the free image processing software "Image Tool" to the heat flux. It is a binary file format that converts each heater into a group of 4x4 pixels. This image can then be processed by a 4x4 smoothing mask to produce a smoothed image of the heat flux.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	

Format	Matrix Files
Description	This format saves each time step as a separate matrix file.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	

Format	Average Heat Flux
Description	Saves a space-averaged heat flux over all 96 heaters for each time step, and appends a total average heat flux for all the time steps to the end.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	

Format	Local Time Average
Description	Saves the time average heat flux of each heater. It also writes an offset file which can be used with the natural convection data to offset other heat flux files.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	.AVG

Format	Local RMS fluctuation
Description	Saves the standard deviation of the heat flux for each heater in the array.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	.RMS

Format	Total Probability Density
Description	<p>Saves a probability density function plot of the heat flux for every heater in each column, and one for the entire array in the last column. An interval of <math>1 \text{ W/cm}^2</math> is used to compute the distribution, and then the distribution is normalized over that interval, and over the total number of heat flux samples, so that the sum of the area under the function is unity.</p> <p>Several other operations can be performed on the probability density distribution to obtain additional information. The mode, median, minimum and maximum percentile limits, and the difference between the minimum and maximum (referred to as the width of the probability distribution), can also be computed.</p>
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name, Probability Width File, Mode File, Median File, Max File, Min File.
Extension	.PDD

Format	FFT
Description	Saves an FFT of each heater in the array. The first column lists the frequency, and the other columns list the magnitude of the frequency-domain component.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name
Extension	.FFT

Format	Boiling Function
Description	Saves a set of data which characterizes the heat flux during boiling events on the surface of the heater. One file contains a map of the heaters which are boiling and not boiling. This file is a binary file with 8-bit record length, where the boiling heaters have a value of 255, and the non-boiling heaters have a value of zero
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name, Boiling Fcn. Time History, Heat Flux Threshold, $q'$ threshold, $q''$ threshold, Boiling Function Stats The other file which is saved by this format, usually with a (.BFS) extension, contains the boiling function statistics. It contains a map of the boiling fraction for each heater in the array, and gives values of the boiling fraction, average heat flux, boiling heat flux, low heat flux and high heat flux (bounded by the percentile limits) for concentric rings of heaters and groups of heaters Another file which is saved is the boiling fraction time history, which gives an average of the boiling function over the entire array.
Extension	.BFN .BFS .BTH

Format	Sample Derivative
Description	This file provides a preview of the boiling function for a particular heater, using the specified boiling function thresholds.
Controls	Input File Name, Output File Name, Offset File (read), Resistance Coefficient, Resistance Constant, Area File Name, Threshold Sample, Heat Flux Threshold, $q'$ threshold, $q''$ threshold.
Extension	.DER

### **B.3.1. "Post-Processing" Window Controls**

Following is a summary of the function of each control in the post-processing windows, shown in Figure B.6.

**Status.** This label indicates the status of the conversion process as a percentage complete.

**File Format.** This box allows the user to select one of several data reduction procedures. These procedures are summarized in one of the following sections. Which of these options is selected determines which of the following items of information must be supplied.

**Input File Name.** The name of a binary input file, ending with ".BIN", on which to perform a data reduction step.

**Output File Name.** The name of an output file. The extension depends on the type of data reduction step being performed. For instance, the file extension for a total probability density distribution file is ".TPD".

**Probability Width File.** A file which expresses the width of the probability density distribution for each heater in the array. The width is determined by the percentile bounds, which are set in the upper and lower percentile limits text boxes.

**Mode File.** A file which contains the statistical mode of the heat flux values seen on that heater for each heater in the array.

**Median File.** A file which contains the statistical median of the heat flux values seen on that heater for each heater in the array.

**Max File.** A file which contains the Upper percentile heat flux value for each heater in the array.

**Min File.** A file which contains the lower percentile heat flux value for each heater in the array

**Boiling Function Statistics.** A file which contains a map of the space-resolved average boiling heat flux, along with the boiling heat flux, boiling fraction, average heat flux, min heat flux and max heat flux for various rings and groups of heaters. It also includes the threshold values that were used for the heat flux and the first and second derivatives.]

**Offset file (write).** A file to which the offset information for natural convection and substrate conduction will be written.

**Boiling Function Time History.** A file containing an array average boiling function time history.

**Max. Number of Steps.** A number indicating the maximum number of time steps that will be processed by the program.

**Number of Heaters.** A number specifying how many of the heaters to perform certain steps on.

**Offset File (read).** A file containing offset information to be read in and used when processing the data files.

**Resistance Coefficient.** A file containing the slope of the resistance vs. Temperature curve for each heater in the array. This information is stored as a calibration file header followed by a series of numbers.

**Resistance constant.** A file containing the y-intercept of the resistance vs. temperature curve for each heater in the array. The resistance coefficient and resistance constant are given so that the temperature in Degrees Celsius is used to convert the temperature to a resistance, which is then used to compute the power dissipation of each heater element.

**Area File Name.** A file which contains the surface area of each of the 96 heater elements. This area is measured by considering a portion of the area between the heaters.

**Heat Flux Threshold.** A value which will separate data points with heat fluxes higher than this into a boiling category, and data points with values lower than this into a non-boiling category.

**q' threshold.** a value which will separate data points with a first derivative higher than this into a boiling category, and all other data points into a non-boiling category.

**q'' threshold.** a value which will separate data points with a second derivative higher than this into a boiling category, and all other data points into a non-boiling category.

**Threshold Sample.** A file which will contain a sample of the time-history, first-derivative, second-derivative and boiling function for one heater. The purpose of this file is for checking the validity of the threshold values.

**Lower Percentile Limit.** the lower percentile bound, used in calculating statistical parameters

**Upper Percentile Limit.** The upper percentile bound, used in calculating statistical parameters.

**Cancel button.** Exit the window

**Convert button.** Begin performing data reduction on a file

**Batch Processing button.** This button opens a window which allows the user to enter a file name which contains information about a batch of data reduction steps.

## **B.4. GRAPH**

The GRAPH program is used to display heat flux data immediately after it has been acquired. Its program window, which is shown in Figure B.7, displays a grid of squares which represent the heaters in the heater array. To the left of the grid, a scale is displayed which relates the colors to the power in mW that the heater is dissipating. Data points which exceed the scale are displayed as white squares. Below the grid is a box which displays the time in ms.

In order to load a file into the graph program, first enter the file name of a binary data file into the File Name box. Double-click on the box in order to browse the files on the disk. Then enter the number of points to be loaded from the file in the Max. Points box. If all the points in the file are to be loaded, enter "0".

Press Load in order to load the data from disk into memory and convert from voltage data to power data. The program will not respond at all while this operation is taking place. When it is completed, press Start to display the data on the grid.

The data file can be displayed one step at a time, or any specified number of steps at a time, by using the Step button. For instance, zero in the Frames box will cause the data file to advance one frame every time the Step button is pressed. Ten in the Frames box will cause the data file to play for 11 frames and stop.

Heat flux offset, resistance and area information is not used to perform the conversion from heater voltage to heater power. The heater resistance is assumed to be 1000 ohms. Since heater power, not heater heat flux, is displayed, then heater size is not used.

#### **B.4.1. "Graph" Window Controls**

Following is a summary of controls in the GRAPH program. These controls are shown in Figure B.7.

**Max. Points.** Limits the maximum number of time steps that are loaded and displayed

**File Name.** The file name of the binary file that is loaded and displayed. Double click on the box to activate a file browsing box.

**Load.** Loads the binary file into memory and converts it from voltage data to heat flux.

**Start.** Begins displaying the heat flux data on the grid one frame after another as fast as the computer hardware will allow.

**Max. Power.** The maximum power level to display. The values between zero and this level are scaled to use the entire color scale, and values above the maximum power level are displayed as white squares.

**Rescale.** Initiates the re-scaling operation on the data in memory.

**Frames.** Number of frames to step forward when the Step button is pressed

**Time.** The time text box displays the time from the starting point in milliseconds.



## **B.5. FILE FORMAT SPECIFICATIONS**

### **B.5.1. Calibration File Format**

In order to modify the calibration file, it is important to know the format of the calibration file. The calibration file is simply a text file, with numbers separated by lines, and with text fields surrounded by quotation marks. The first part of the file contains information about the setup configuration that was used to make the calibration file. The second part of the file contains the actual calibration data.

1. Threshold voltage, mV
2. No longer used
3. Voltage step rate, Steps/s
4. No longer used
5. DeltaV, mV
6. ADC Offset
7. Scans per datapoint
8. Hardware Scanrate
9. Gain index - No longer used
10. Heater Method - 0 = All, 1 = Range, 2 = Specific
11. First Heater in Range
12. Last Heater in Range
13. Number of heaters in Specific Heaters box
14. Heater numbers for specific heaters. The above field defines how many there will be. If the above field is zero, then skip this field.
15. Comments field, surrounded by quotes.

This concludes the calibration setup data. The following fields consist of temperature-specific information.

1. Temperature
2. Date code
3. Comments
4. List of  $V_{\text{cmd}}$  values, in mV. The number of values corresponds to the number of heaters that are specified above.

### **B.5.2. Binary Data File Format**

The data points are saved in the binary file as 16 bit numbers. DAQBook data files encode a 10 V voltage range as an unsigned 16 bit integer. The custom A/D board encodes a 12 V range as an unsigned 16 bit integer. The conversion from an unsigned integer to a floating point number depends on the A/D hardware type constant that is stored in the .TAG file. A factor of 10 or 12 is used in converting to a floating point number, depending on which hardware type is detected.

### **B.5.3. Control Setup File Format**

The setup file for the control program is typically saved using a .CTL extension. It contains the following fields

1. The first 16 lines contain the file names of the calibration files
2. This variable is not presently used
3. Gain Index - Not presently used
4. A/D sampling rate
5. A/D sampling duration
6. Delay time - Not presently used

7. Trigger source - Not presently used
8. Temperature
9. Method for selecting heaters to sample- 0 = All, 1 = Range, 2 = Specific
10. First heater in range
11. Last heater in range
12. Number of heaters in Specific Heaters box
13. Heater numbers for specific heaters. The above field defines how many there will be. If the above field is zero, then skip this field.
14. Data file format
15. Comments from setup
16. Setup file name
17. A/D hardware type
18. D/A hardware type
19. A/D Channel Offset

#### **B.5.4. Tag File Format**

The tag file typically has a .TAG extension. It is a text file that must be attached to the data file because the data file contains no information about how to interpret the binary file. It contains the CONTROL setup file format information listed above, and some information that is specific to a particular data file. It follows a format similar to the calibration file, with numbers on separate lines, and text fields surrounded by quotation marks.

1. The first part of the file contains the same information listed in the CONTROL setup file format information above.
2. Comments from the main CONTROL window

3. Date field

4. Time field

Several other files are necessary to perform the conversion from heater voltage to surface heat flux. A description of these files and their format is given here.

#### **B.5.5. Resistance File Format**

In order to calculate the heat flux more accurately from the heater voltage, two files containing heater resistance information are loaded. The first file contains information on the slope of the resistance vs. Temperature line, and the second file contains information on the y-intercept of this line. The equation  $y=mx+b$  is used to calculate the resistance, where  $y$  is the calculated resistance,  $x$  is the temperature,  $m$  is the slope and  $b$  is the  $y$  intercept.

The format of both of these files is the same as the calibration files, except that instead of  $V_{cmd}$  values, slope information is saved in one file and  $y$ -intercept information in the other.

#### **B.5.6. Offset File Format**

The offset files contain the substrate conduction heat flux for each heater at a given temperature. These values are subtracted from the heat flux values that are calculated from the binary data files to obtain the value of the heat flux leaving the top of the heater elements. Thus, a different offset file must be specified for each heater temperature. The format for this file is the same as the calibration file format, except that instead of a  $V_{cmd}$  value, it contains a list of heat fluxes. The important quantities in this file are the temperature and the offset values. The offsets are given in units of  $W/cm^2$ .

#### **B.5.7. Size File Format**

In order to calculate the heat flux more accurately, the area of each heater element, including a portion of the unheated surface between the heaters, is stored in this file. The format

of this file is also the same as the calibration file format, except that instead of  $V_{cmd}$  values, the file contains a list of heater areas, in units of  $\text{cm}^2 10^4$ . That is, the values in the file must be multiplied by  $10^{-4}$  to obtain units of  $\text{cm}^2$ .

## B.6. FIGURES

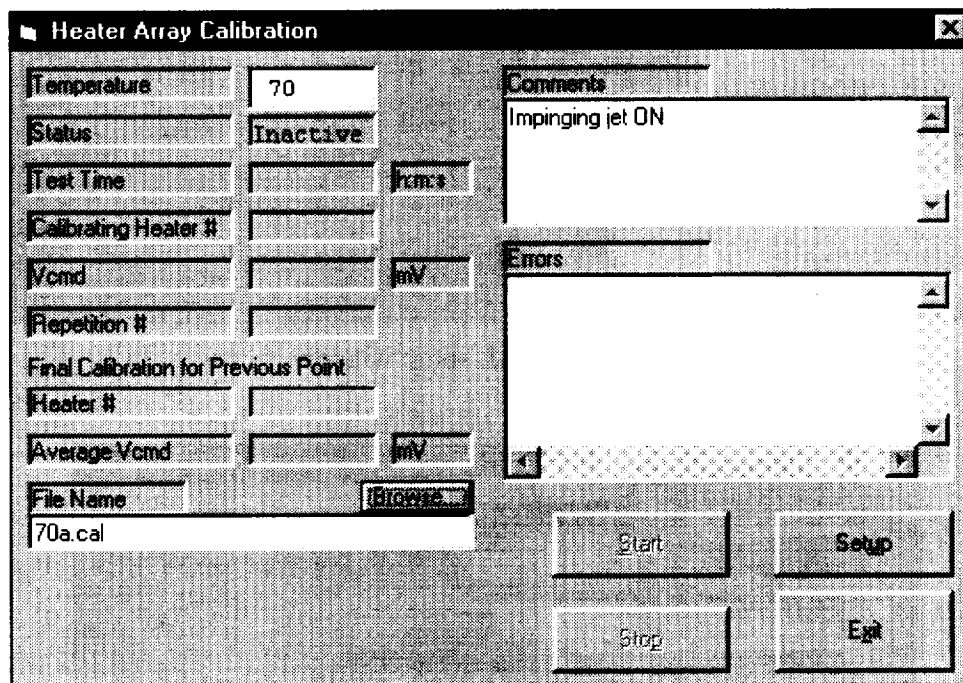
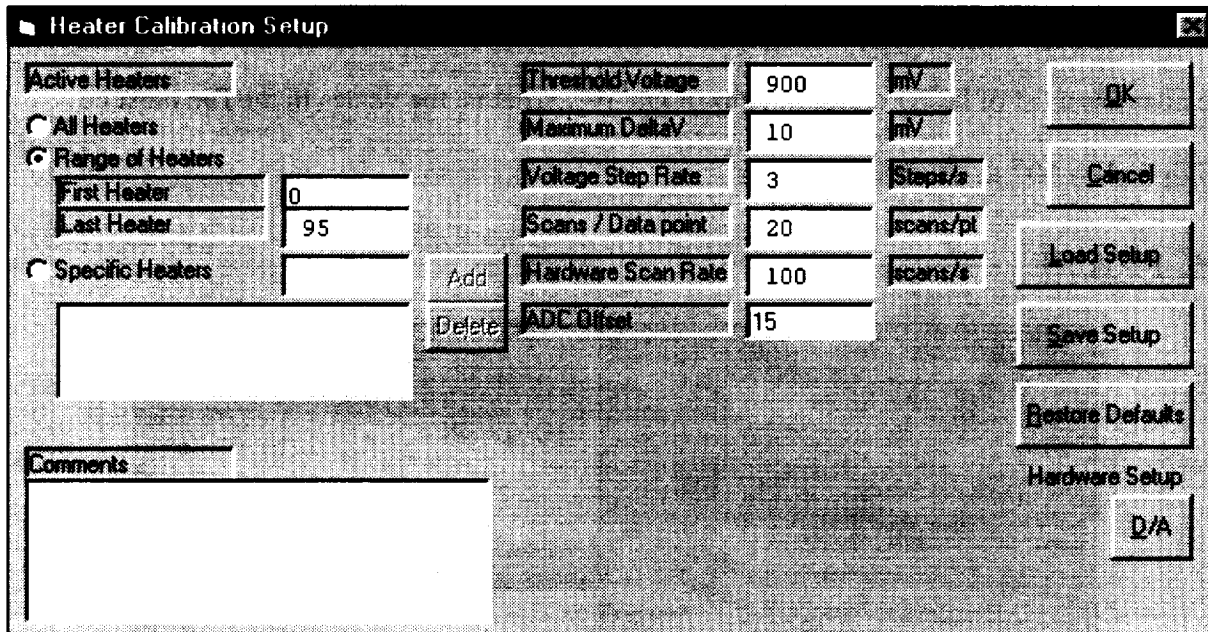


Figure B.1: CAL window



**Heater Calibration Setup**

**Active Heaters**

☐ All Heaters

☒ Range of Heaters

First Heater: 0

Last Heater: 95

☐ Specific Heaters

Add

Delete

Threshold Voltage: 900 mV

Maximum Delta V: 10 mV

Voltage Step Rate: 3 Steps/s

Scans / Data point: 20 scans/pt

Hardware Scan Rate: 100 scans/s

ADC Offset: 15

OK

Cancel

Load Setup

Save Setup

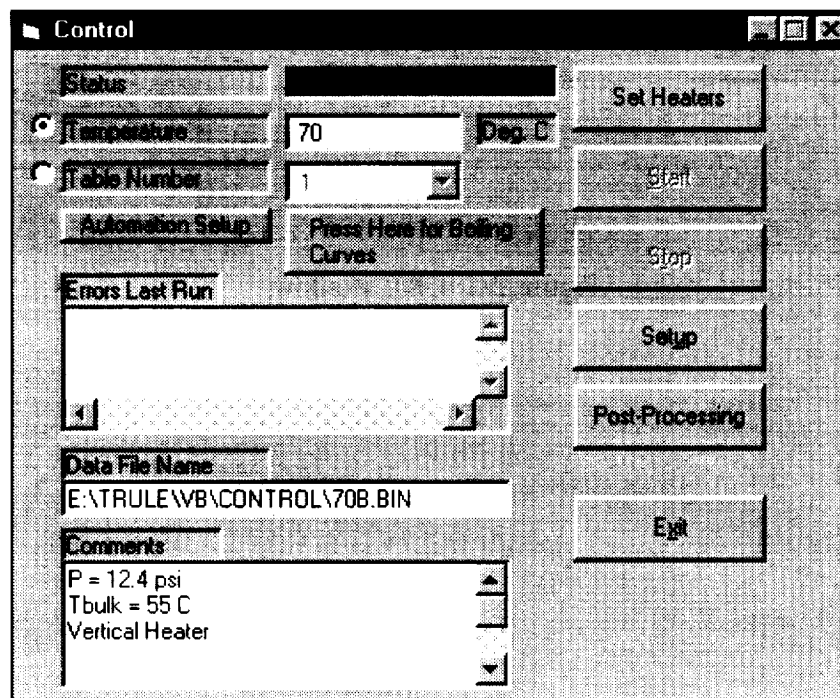
Restore Defaults

Hardware Setup

D/A

Comments

Figure B.2: CAL setup window



**Control**

Status: [Blank]

☒ Temperature: 70 Deg. C

☐ Table Number: 1

Automation Setup

Press Here for Boiling Curves

Errors Last Run

Data File Name: E:\TRULE\WB\CONTROL\70B.BIN

Comments: P = 12.4 psi  
Tbulk = 55 C  
Vertical Heater

Set Heaters

Start

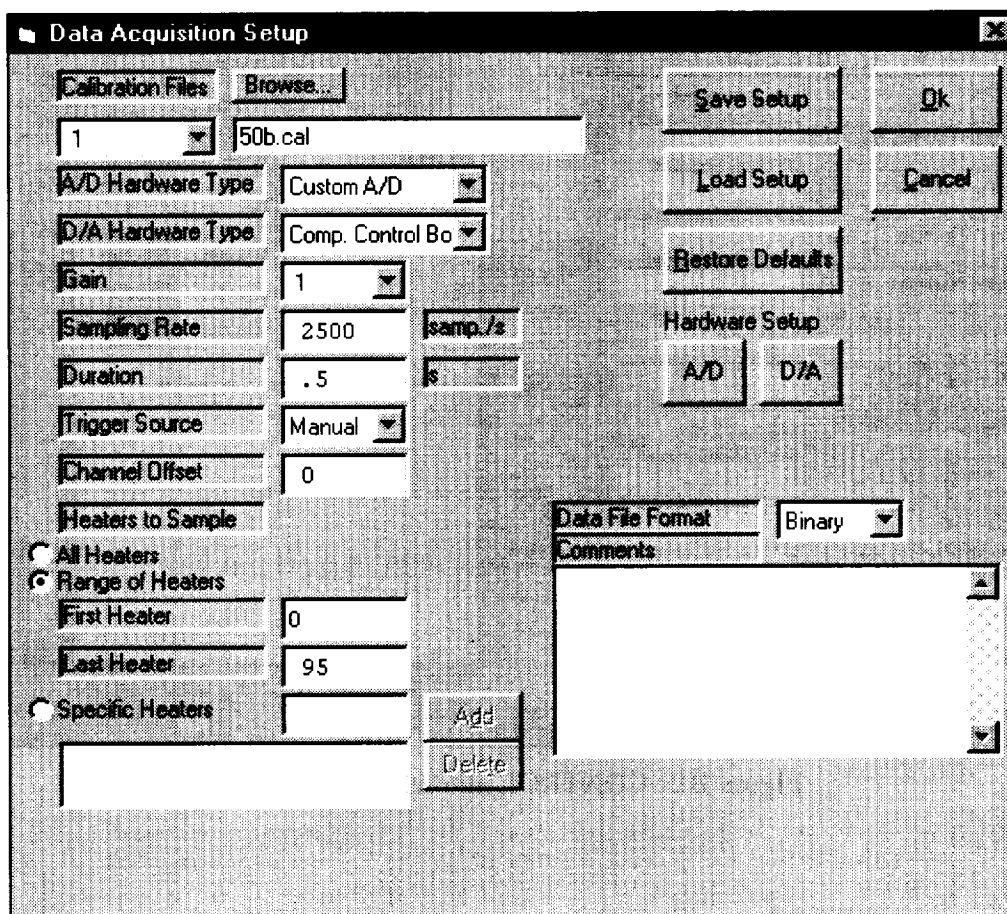
Stop

Setup

Post-Processing

Exit

Figure B.3: CONTROL window



The image shows a 'Data Acquisition Setup' window with various configuration options. The window is titled 'Data Acquisition Setup' and has a standard Windows-style title bar with a close button. The main area is divided into several sections. On the left, there's a 'Calibration Files' section with a 'Browse...' button and a list showing '1' and '50b.cal'. Below this are 'A/D Hardware Type' (set to 'Custom A/D'), 'D/A Hardware Type' (set to 'Comp. Control Bo'), 'Gain' (set to '1'), 'Sampling Rate' (set to '2500' with units 'samp./s'), 'Duration' (set to '.5' with units 's'), 'Trigger Source' (set to 'Manual'), and 'Channel Offset' (set to '0'). The 'Heaters to Sample' section has three radio buttons: 'All Heaters', 'Range of Heaters' (which is selected), and 'Specific Heaters'. Under 'Range of Heaters', there are fields for 'First Heater' (set to '0') and 'Last Heater' (set to '95'). Below these are 'Add' and 'Delete' buttons. On the right side, there are buttons for 'Save Setup', 'Load Setup', 'Restore Defaults', and 'Hardware Setup'. The 'Hardware Setup' section has two sub-buttons: 'A/D' and 'D/A'. At the bottom right, there's a 'Data File Format' dropdown set to 'Binary' and a 'Comments' text area with a scroll bar. The window also has 'Ok' and 'Cancel' buttons in the top right corner.

Section	Parameter	Value
Calibration Files	File Name	50b.cal
	Index	1
A/D Hardware	Hardware Type	Custom A/D
	Gain	1
D/A Hardware	Hardware Type	Comp. Control Bo
	Sampling Rate	2500 samp./s
Timing	Duration	.5 s
	Trigger Source	Manual
Channel Settings	Channel Offset	0
	Heaters to Sample	Range of Heaters
Heater Range	First Heater	0
	Last Heater	95
Data File	Data File Format	Binary
	Comments	

Figure B.4: CONTROL setup window

**Data Acquisition Automation**

Task: Data Acquisitor [Delete] [Copy Down]

Base File Name: [ ]

First Number: [ ]

First Number: [ ]

Last Number: [ ]

Number Format: Don't Fill [v]

Number Length: [ ]

Temperature Mode: Manual Entry [v]

Lowest: [ ]

Highest: [ ]

Interval: [ ]

Delay Time: 10

Auto File Name: E:\TRULE\CFG\W

[Load Automation] [Save Automation] [Update Column] [START]

Edit Cell: 60b.bin

	Temper- ature	Input File	DeadBook File	High-speed file
1	50		50a.bin	50b.bin
2	55		55a.bin	55b.bin
3	60		60a.bin	60b.bin
4				
5				
6				
7				
8				
9				
10				
11				
12				

Figure B.5: CONTROL automation window



**Post-Processing**

Status	Ready	Max. Number of Steps	
File Format	Boiling Function	Number of Heaters	
Input File Name	E:\TRULE\DAT\082397\0823_	Offset File (read)	E:\TRULE\DAT\082397\0823_
Output File Name	E:\TRULE\DAT\082397\0823_	Resistance Coefficient	E:\TRULE\CFG\0828A.RES
Probability Width File	E:\TRULE\DAT\082397\0823_	Resistance Constant	E:\TRULE\CFG\0828B.RES
Mode File	E:\TRULE\DAT\082397\0823_	Area File Name	E:\TRULE\CFG\0620A.SIZ
Median File	E:\TRULE\DAT\082397\0823_		
Max File	E:\TRULE\DAT\082397\0823_	Heat Flux Threshold	12
Min File	E:\TRULE\DAT\082397\0823_	q' Threshold	2500
Boiling Function Stats	E:\TRULE\DAT\082397\0823_	q'' Threshold	1.5E7
Offset File (write)		Threshold Sample	
Boiling Fcn. Time Hist.	E:\TRULE\DAT\082397\0823_	Lower Percentile Limit	5
		Upper Percentile Limit	5

Figure B.6: CONTROL post-processing window

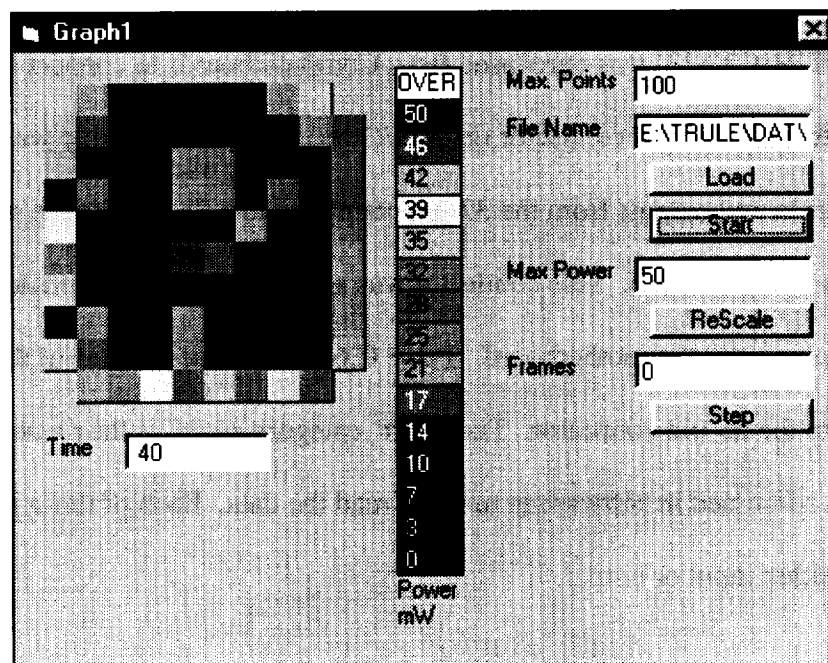


Figure B.7: GRAPH window

## **APPENDIX C: CUSTOM A/D SYSTEM**

This appendix describes the design and operation of the custom A/D board that is used to collect heat flux data at very high speeds. Section C.1. describes the physical connections to the board, and how the I/O operations are programmed. Section C.2. describes the operation of the individual components on the board.

### **C.1. CONNECTIONS AND I/O OPERATIONS**

#### **C.1.1. Cabling**

The custom A/D system connects to the computer through a 25-pin male sub-D connector on the motherboard. The computer interfaces to the A/D system using a digital I/O adapter, the PCM-D24CTR from Computerboards, Inc. This device provides 24 channels configurable as either inputs or outputs. The Computerboards universal library is used to address the board in Microsoft Visual Basic.

A cable (PCM-C37/33) was purchased from Computerboards to connect a female 37-pin sub-D connector to a specialized connector on the I/O board. A second cable was fabricated to map the digital inputs and outputs from the 37-pin connector to a 25-pin female sub-D connector which connects to the A/D board. This connector was adapted to a 25-pin female sub-D connector which plugs into the motherboard. Table C.6 shows how the pins of the 37-pin connector map into the 25-pin connector. The "Port" category contains the name of the input or output port that is addressed in software in order to read the data. Each of these ports can be programmed for either input or output.

	Input or Output	25-pin	37-pin	Port
Group Address	Output	19-22	1-4	FirstportA
Frequency Select	Output	18,17	5,6	FirstportA
RESET	Output	13	7	FirstportA
Increment	Output	14	8	FirstportA
Data Values	Input	1-8	9-16	FirstportB
Data Values	Input	9-12	17-20	FirstportCL
DONE	Input	15	21	FirstportCH
START	Output	16	22	FirstportCH
Ground		23-25	33	

Table C.6: List of A/D Cable Connections

Table C.7 lists the function of the test-point pins which are located along the edge of the board. These pins can be used to check vital functions when troubleshooting.

Test Point #	Function
TP1	+ 10 Volt voltage reference
TP2	D/A Converter Output (Vcmd)
TP3	Pulse Signal

Table C.7: Test Point Functions

### **C.1.2. Data acquisition and downloading procedure**

Each A/D card holds two distinct A/D systems. Each system digitizes 16 channels, and stores the digitized values in memory. Each of these memory segments is addressed as a separate "group" when the data is downloaded from the A/D card. Therefore, if 96 heaters are being digitized, it will be necessary to address 6 groups, groups 0 through 5, in order to download all the values. Table C.8 relates group address to which A/D card and which feedback card will be addressed.

Group Address	A/D Card Number	Feedback Card Number
0	1	1
1	1	2
2	2	3
3	2	4
4	3	5
5	3	6
6	4	7
7	4	8
8	5	9
9	5	10

Table C.8: Group Address Reference

The sampling frequency is set by setting the frequency select value. Table C.9 lists the frequency select bit values and the corresponding sampling frequency.

Sampling Frequency (samples/s)	Register value
10,000	0
5,000	1
2,500	2
1,250	3

Table C.9: Frequency Select Register Values

The following procedure is implemented by the Visual Basic program to control the A/D board. Note that "pulse" means the signal goes from LOW to HIGH back to LOW.

To Collect  
Data:

1. Pulse RESET signal
2. Set Frequency Select
3. Pulse RESET
4. When DONE goes HIGH, begin data reading sequence

To Read Data:

1. Set group address
2. Pulse RESET signal
3. Read and store 12 bit word
4. Pulse Increment Address
5. When DONE goes HIGH, all data has been read from this group. Proceed to next group by returning to step 1. If the last group has been read, then quit.
6. Otherwise, return to step 3 to continue reading data.

## **C.2. DETAILS OF A/D BOARD DESIGN**

### **C.2.1. Introduction**

This section describes in greater detail the design of the high-speed A/D system that was designed at the University of Denver by Mr. Richard Quine for acquiring data from the constant-temperature heater array at very high speeds. A schematic for the high-speed A/D board is included in Appendix F.3. The following text will refer often to the timing diagram in Figure C.1 for clarity.

The first and the central part of the design is the A/D chip. Typical A/D chips require an external sample and hold chip to stabilize the voltage that is being digitized. However, this particular A/D chip, the MAX122 from Maxim, has sample and hold circuitry built into the chip itself. After digitization is finished, the 12-bit digital values will be stored sequentially in a memory chip. Finally, the board will communicate and transfer the data stored in memory to the laptop computer through a third party PCMCIA digital I/O card.

This A/D chip utilizes successive approximation to convert an analog voltage into a 12-bit digital word. 13 clock cycles after digitization begins, the digital data is made available on the

output bus of the chip. As a requirement of this research project, the A/D system must sample data at a rate as high as 10,000 samples/s/channel. Since each A/D chip will sample 16 channels that have been multiplexed down into one single analog input to the A/D chip, the A/D chip has to operate at a clock speed of 2.72 MHz in order to achieve an effective sampling rate of 10,000 samples/sec/channel. In this design, the sampling rate is software adjustable using *F1* and *F0* signals. Counter (U4A) is used to divide the clock down by factors of 2 (LSB), 4, 8, and 16 (MSB). Therefore the clock used runs at 5.44 MHz. Using this clock speed, sampling rates of 10000, 5000, 2500, and 1250 samples/s/channel are achieved for the given factors.

Following is a detailed calculation of the clock frequency requirements of the A/D board.

Each channel requires 10 kilosamples/sec/channel

$$f = 10\text{-KHz}$$

Each A/D chip samples 16 channels.

$$N = 16$$

Total frequency for A/D converter that serves 16 channel is calculated.

$$f_{\text{total}} = N \cdot f$$

$$f_{\text{total}} = 160\text{-KHz}$$

Total conversion time is calculated.

$$\tau = \frac{1}{f_{\text{total}}}$$

$$\tau = 6.25 \cdot 10^{-6} \cdot \text{sec}$$

From the timing diagram, Figure C.1, it takes a total of 13+4=17 clock cycles to achieve a conversion.

$$\tau_{\text{clock}} = \frac{\tau}{17}$$

Time per clock cycle

$$\tau_{\text{clock}} = 3.676 \cdot 10^{-7} \cdot \text{sec}$$

The clock frequency required to achieve 10 kilosamples/sec/channel is calculated.

$$f_{\text{clock}} = \frac{1}{\tau_{\text{clock}}}$$

$$f_{\text{clock}} = 2.72 \text{ MHz}$$

Static random-access-memory (SRAM) from Cypress is being used to store the data before it is transferred to the computer.

The CY7C106A is a 256K\*4 high performance CMOS SRAM. Since each chip is only 4 bits wide, three of the CY7C106A are used to store 12 bits of data. In other words, using three CY7C106A chips, the board can store up to 256 Ksamples of data before it is downloaded to the computer.

In order to ensure clarity, italicized words indicate signals, and the forward slash, “/”, is the notation for an inverted signal. For example,  $/B$  is the inverted  $B$ . Confusion can be minimized by constantly referring to the circuit schematic in Appendix F and to Figure C.1, the timing diagram.

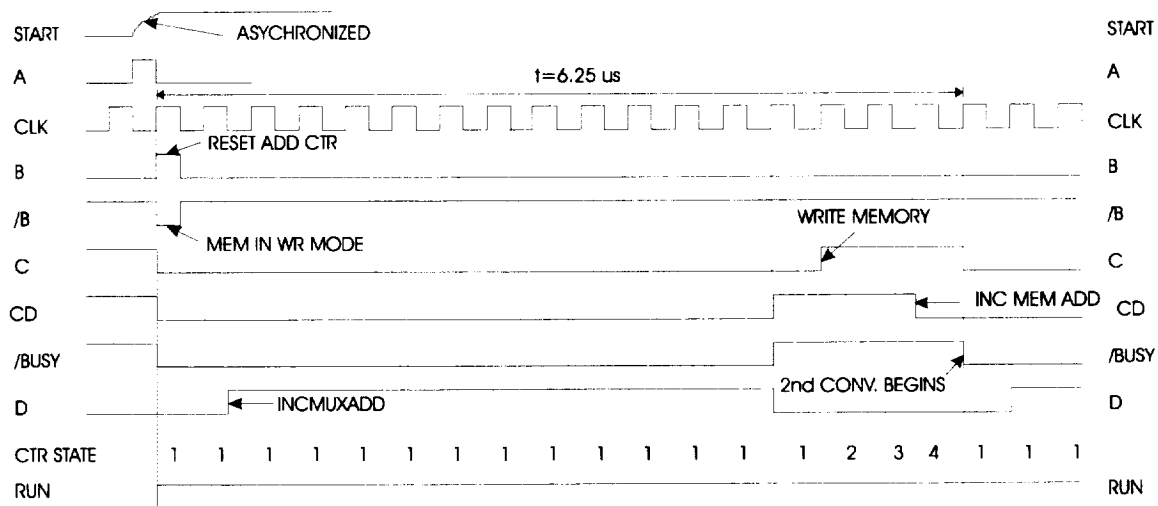


Figure C.1a: Begin conversion

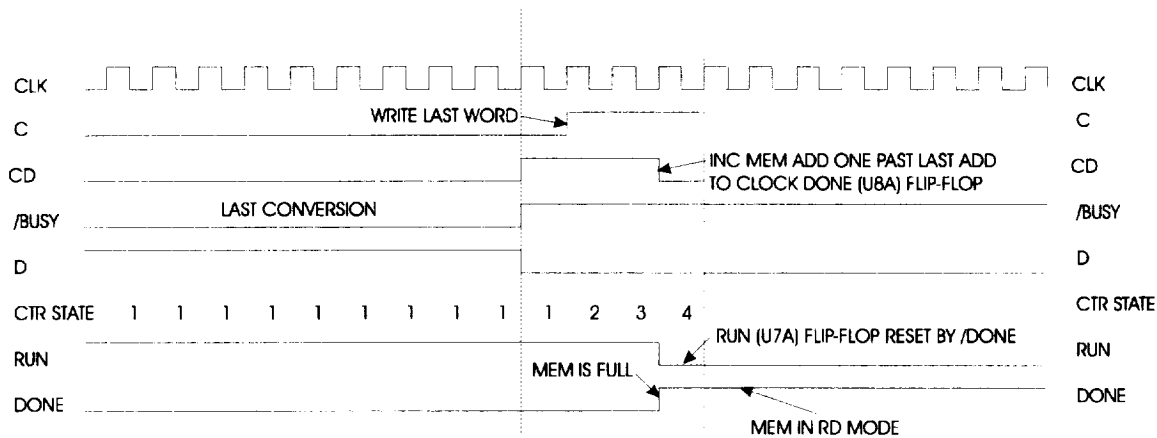


Figure C.1b: Last Conversion

Figure C.1: Timing Diagram of A/D System

### **C.2.2. Analog to Digital Converter Chip Control Signals**

To initiate data acquisition, an external *START* signal is sent from the computer through PCMCIA. In order to follow the explanations below, constant referral to the A/D system circuit schematic in Appendix F is extremely helpful. The asynchronous *START* signal will cause the D-input in flip-flop A (U6A) to appear in the output Q. The *CLK* will then synchronize signal A on the positive going edge and output it in flip-flop B (U6B). As soon as */B* becomes low (one *CLK* cycle after *START*), it sets flip-flop RUN (U7A) to high and resets flip-flop A (U6A) and C to low. Output of flip-flop C (U7B) and */RUN* both being low will activate the */RD* and */CS* on the MAX122 and hence begin acquisition. Since A has been reset, */B* becomes high in the next *CLK* cycle. When */B* becomes high, flip-flop C (U7B) is not held reset and flip-flop RUN (U7A) is not held set anymore. *RUN* will stay high as long as *DONE* or *PCRES* stays low. *DONE* will go high only when the memory is all filled and ready to be transferred to the computer. XOR gate with one of the inputs tied to high +5V will act as an inverter. Before conversion, */BUSY* signal is high, therefore flip-flop D (U21A) has been reset to low. *BUSY* and */RUN* is high, so the output of the NOR gate (U11A) is low, so the output of counter (U10) kept loading 0001<sub>2</sub>. Q<sub>C</sub> is low



and */BUSY* is high, output of the XOR gate, *CD* is high. During conversion, */BUSY* signal will become low until conversion is finished and the 12-bit data is available on the output pins. */BUSY* being low will not reset flip-flop D (U21A) anymore, therefore in the next *CLK* cycle, the output of D goes high until */BUSY* goes low again. The positive edge of *D* will increment the multiplexer address *INCMUXADD*. Also with */BUSY* low, *CD* will be low until */BUSY* goes high that signifies conversion completes. The conversion will take 13 *CLK* cycles. Now that the conversion is finished, */BUSY* and *CD* are both high and *RUN* is still high, the output of the NOR gate (U11A) will turn into a high. The counter (U10) will begin counting from the next *CLK* cycle. After CTR counts to  $0100_2$  ( $4_{10}$ ), the *CD* signal will go from high to low. In the transition from high to low in *CD*, the memory address will be incremented by *CD*. Note that output of flip-flop C (U7B) depends upon *CD*, so when *CD* goes low as */BUSY* remains high (after conversion), the next clock cycle will cause data acquisition to start all over again. The digitized voltage is output in a 12-bit data labeled *MDA0* being the LSB to *MDA11* being the MSB.

### **C.2.3. Multiplexer Control Signals**

Before the first conversion, */RUN* is high and therefore the output of the 4-Stage Binary Ripple Counter (U13A) is  $0000_2$ . The *INCMUXADD* signal is fed into the counter and every negative-going edge will advance the counter by one state. The 4-bit multiplexer address corresponds to one of the 16 analog inputs that is to be digitized. The output of the multiplexer is conditioned by analog circuits before entering the MAX122.

### **C.2.4. Memory Control Signals**

The memory chip can operate in either write or read mode. The CY7C106A is in write mode when the MAX122 is transferring data into the memory. Conversely, read mode occurs

when data is being transferred from the CY7C106A to computer. Write mode is controlled by signal */WR* driven by *C* and *INCMEMADD* driven by *CD*. In read mode, */RDA* or */RDB* and *ADDINC* from PCMCIA serve as the control signals. One A/D board has two groups of A/D system. Each group serves 16 channels. To differentiate between groups, A and B refers to each individual group on the same board. In write mode, every group will write concurrently. Only in read mode do different groups need to be separated. This explains the reason for having */RDA* and */RDB*.

In the write mode, conversion has just finished. 12-bit data is ready to be written to memory on *MDA0* to *MDA11*. Flip-flop W/R (U8B) is set by */B* being low when the first conversion is initiated. This brings the S (select) pin on the data selector (U20) to low and 1A, 2A, 3A, and 4A will be transferred to 1Y, 2Y, 3Y, and 4Y. *B* being high for one *CLK* cycle will reset the address counters (U17A, U17B, U18A, U18B). Each negative going edge of *INCMEMADD* driven by *CD* will advance the counter (U17A) by one state. This will increment the memory address after the previous data has been written. */WR* driven by *C* changes from low to high one *CLK* cycle after *CD* goes high. */WR* going high will write data into memory. 18 bits are needed to address 256K memory. When the memory is filled, the 19<sup>th</sup> bit on the counter (*QC* in U13B) turns into a high. This will change the output of flip-flop DONE (U8A) to high. This *DONE* signal informs the computer that the memory is filled with data and ready to be read.

From the previous paragraph, */DONE* turns low when memory is ready to be read. */DONE* will reset the flip-flop W/R (U8B), so S on data selector (U20) will turn high. Similar to the write cycle, 1Y will be driven by *ADDINC* from the computer interface, 2Y is */RDB* driven by *GSELB*, 3Y is high, and 4Y is */RDA* driven by *GSELA*. *GSELA* or *GSELB* is selected by the computer interface when reading the data to the computer. A *PCRES* signal is sent from the

computer to clear the memory address before data is read. Now the address counters are all lows. In read mode, the increment address signal is controlled by the computer. */RDA* will enable the tri-state buffers for group A (U22A, U22B, U23A) to take control of the data bus to the computer. */RDB* will enable the tri-state buffers for group B (U24A, U24B, U23B). Meanwhile, */RDA* will enable data to be put on the data bus to be stored into the computer. After all data has been read from the memory, again when the 19<sup>th</sup> bit on the counter ( $Q_C$  in U13B) turns high, *DONE* will change to high to inform the computer.

### **C.2.5. Conditioning Analog Circuits**

MAX122 is a bipolar A/D converter which accepts voltages from -5V to +5V. Since the input analog signal will vary from 0V to 12V, it needs to be offset and attenuated before entering the MAX122. Op-Amp LF347 (U28A) with R1 and R2 will attenuate the voltage range of 12V to approximately 10V. R3, R4, and R7 is to offset the input voltage.

## APPENDIX D: CONNECTIONS BETWEEN COMPONENTS

A 15 pin male connector on the motherboard is used to supply power to the control system. Table D.1 describes the pin assignment of this power connector. The same connector is located on the decoding board, and the pin assignments are identical, except that pins 1,2,5, and 6 are not connected. +5V power and +24V power are not needed on the decoding board.

Pin	Function
1,2	+5V
3,4	+15V
5,6	+24V
7,8	-15V
9,10	-5V
11,12,13,14,15, 16	GND

Table D.1: Pin assignment for power connector on motherboard.

A 9 pin female connector on the motherboard is used to connect a personal computer to the microprocessor control board using an RS-232 serial port connection. Table D.2 describes the pin assignment of the RS-232 connector.

Pin	Function
2	RXD
3	TXD
4	DTR
8	CTS/DSRD CD
5	GND

Table D.2: Pin assignment for RS-232 connector on motherboard

The 26-pin connectors on the motherboard are used to connect the feedback boards to the decoding board. Ribbon cables are used to make this connection. Table D.3 lists the pin assignment of the ribbon-cable connectors on the motherboard and the decoding board. Each connector carries 16 heater power leads and 10 ground leads. Originally, this large number of ground leads was specified so that it would provide a very small voltage drop between the

motherboard ground and the decoding board ground bus bar. However, in practice the difference can be significant when one to two amps is flowing between the motherboard and the decoding board.

Pin	Function
1	Feedback board voltage-sensing lead
2-5	GND
6-21	Heater power lead connections
22-26	GND

Table D.3: Pin assignments for 26-pin ribbon cable connectors

Instead, the ground voltage controller circuit on the decoder board is used to maintain a steady ground voltage. In order for this to work properly, several wires in the ribbon cable must be cut. Doing so increases the resistance in the current path between the ground on the feedback boards and the grounding bar on the decoding board. This is necessary because the control circuit on the decoding card is designed to draw the voltage of the ground bar down to a level that can be several mV lower than the ground plane of the motherboard. If the current path between the ground on the motherboard and the ground on the decoder board is too low in resistance, then the control circuit would have to sink a very large current in order to maintain the proper ground voltage.

Table D.4 describes the pin assignment of the 26-pin ribbon-cable connections. The ribbon cables should connect pins of the same number on each end of the cable.

Pin	Function
1	Feedback board voltage-sensing lead
2-5	No connection
6-21	Heater power lead connections
22-25	No connection
26	Ground

Table D.4: Modified pin assignments for 26-pin ribbon cable connectors

50-pin ribbon cable connectors are used to connect the decoding board to the circuit board that holds the heater array. With the present heater array layout, each pair of adjacent wires on the ribbon cable connects to an individual heater. All the wires are used except the first two, pins one and two, of each cable. Table D.5 illustrates the arrangement of the ribbon cable connector pins.

1	3	5	7	9	11	13	15	17	19	....	47	49
2	4	6	8	10	12	14	16	18	20	....	48	50

Table D.5: Arrangement of ribbon cable connector pins.

Since each adjacent pair of pins, such as 3 and 4, 5 and 6, etc, connects to a heater, all the pins on one side are assigned as ground pins, and are connected to the ground bar. Meanwhile all the pins on the other side are assigned as power leads, and wire-wrap connections are made from that pin to one of the heater leads from the feedback boards.

## **APPENDIX E: DATASHEETS FOR KEY PARTS**

The following pages contain reproductions of manufacturer data sheets for key parts in the control system. These data sheets contain information on offset voltage, offset drift, power supply levels and frequency response that are important when designing the control system.



## Low Cost Analog Multiplier

### AD633

#### FEATURES

Four-Quadrant Multiplication  
Low Cost 8-Pin Package  
Complete—No External Components Required  
Laser-Trimmed Accuracy and Stability  
Total Error Within 2% of FS  
Differential High Impedance X and Y Inputs  
High Impedance Unity-Gain Summing Input  
Laser-Trimmed 10 V Scaling Reference

#### APPLICATIONS

Multiplication, Division, Squaring  
Modulation/Demodulation, Phase Detection  
Voltage-Controlled Amplifiers/Attenuators/Filters

#### PRODUCT DESCRIPTION

The AD633 is a functionally complete, four-quadrant, analog multiplier. It includes high impedance, differential X and Y inputs and a high impedance summing input (Z). The low impedance output voltage is a nominal 10 V full scale provided by a buried Zener. The AD633 is the first product to offer these features in modestly priced 8-pin plastic DIP and SOIC packages.

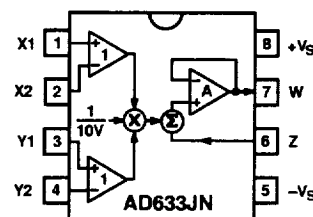
The AD633 is laser calibrated to a guaranteed total accuracy of 2% of full scale. Nonlinearity for the Y-input is typically less than 0.1% and noise referred to the output is typically less than 100  $\mu\text{V rms}$  in a 10 Hz to 10 kHz bandwidth. A 1 MHz bandwidth, 20 V/ $\mu\text{s}$  slew rate, and the ability to drive capacitive loads make the AD633 useful in a wide variety of applications where simplicity and cost are key concerns.

The AD633's versatility is not compromised by its simplicity. The Z-input provides access to the output buffer amplifier, enabling the user to sum the outputs of two or more multipliers, increase the multiplier gain, convert the output voltage to a current, and configure a variety of applications.

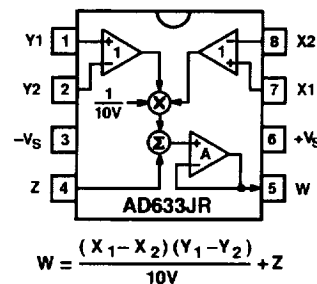
The AD633 is available in an 8-pin plastic mini-DIP package (N) and 8-pin SOIC (R) and is specified to operate over the 0°C to +70°C commercial temperature range.

#### CONNECTION DIAGRAMS

8-Pin Plastic DIP (N) Package



8-Pin Plastic SOIC (R) Package



#### PRODUCT HIGHLIGHTS

1. The AD633 is a complete four-quadrant multiplier offered in low cost 8-pin plastic packages. The result is a product that is cost effective and easy to apply.
2. No external components or expensive user calibration are required to apply the AD633.
3. Monolithic construction and laser calibration make the device stable and reliable.
4. High (10 M $\Omega$ ) input resistances make signal source loading negligible.
5. Power supply voltages can range from  $\pm 8\text{ V}$  to  $\pm 18\text{ V}$ . The internal scaling voltage is generated by a stable Zener diode; multiplier accuracy is essentially supply insensitive.

#### REV. A

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.  
Tel: 617/329-4700 Fax: 617/326-8703



# AD633—SPECIFICATIONS ( $T_A = +25^\circ\text{C}$ , $V_S = \pm 15\text{ V}$ , $R_L \geq 2\text{ k}\Omega$ )

Model		AD633J			
TRANSFER FUNCTION		$W = \frac{(X_1 - X_2)(Y_1 - Y_2)}{10\text{ V}} + Z$			
Parameter	Conditions	Min	Typ	Max	Unit
<b>MULTIPLIER PERFORMANCE</b>					
Total Error	$-10\text{ V} \leq X, Y \leq +10\text{ V}$		$\pm 1$	$\pm 2$	% Full Scale
$T_{\text{MIN}}$ to $T_{\text{MAX}}$			$\pm 3$		% Full Scale
Scale Voltage Error	$SF = 10.00\text{ V Nominal}$		$\pm 0.25\%$		% Full Scale
Supply Rejection	$V_S = \pm 14\text{ V to } \pm 16\text{ V}$		$\pm 0.01$		% Full Scale
Nonlinearity, X	$X = \pm 10\text{ V}, Y = +10\text{ V}$		$\pm 0.4$	$\pm 1$	% Full Scale
Nonlinearity, Y	$Y = \pm 10\text{ V}, X = +10\text{ V}$		$\pm 0.1$	$\pm 0.4$	% Full Scale
X Feedthrough	Y Nulled, $X = \pm 10\text{ V}$		$\pm 0.3$	$\pm 1$	% Full Scale
Y Feedthrough	X Nulled, $Y = \pm 10\text{ V}$		$\pm 0.1$	$\pm 0.4$	% Full Scale
Output Offset Voltage			$\pm 5$	$\pm 50$	mV
<b>DYNAMICS</b>					
Small Signal BW	$V_O = 0.1\text{ V rms}$		1		MHz
Slew Rate	$V_O = 20\text{ V p-p}$		20		V/ $\mu\text{s}$
Settling Time to 1%	$\Delta V_O = 20\text{ V}$		2		$\mu\text{s}$
<b>OUTPUT NOISE</b>					
Spectral Density	$f = 10\text{ Hz to } 5\text{ MHz}$		0.8		$\mu\text{V}/\sqrt{\text{Hz}}$
Wideband Noise	$f = 10\text{ Hz to } 10\text{ kHz}$		1		mV rms
			90		$\mu\text{V rms}$
<b>OUTPUT</b>					
Output Voltage Swing		$\pm 11$			V
Short Circuit Current	$R_L = 0\ \Omega$		30	40	mA
<b>INPUT AMPLIFIERS</b>					
Signal Voltage Range	Differential	$\pm 10$			V
	Common Mode	$\pm 10$			V
Offset Voltage X, Y			$\pm 5$	$\pm 30$	mV
CMRR X, Y	$V_{\text{CM}} = \pm 10\text{ V}, f = 50\text{ Hz}$	60	80		dB
Bias Current X, Y, Z			0.8	2.0	$\mu\text{A}$
Differential Resistance			10		M $\Omega$
<b>POWER SUPPLY</b>					
Supply Voltage			$\pm 15$		V
Rated Performance				$\pm 18$	V
Operating Range		$\pm 8$			V
Supply Current	Quiescent		4	6	mA

## NOTES

Specifications shown in **boldface** are tested on all production units at electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in **boldface** are tested on all production units.

Specifications subject to change without notice.

## ABSOLUTE MAXIMUM RATINGS<sup>1</sup>

Supply Voltage	$\pm 18\text{ V}$
Internal Power Dissipation <sup>2</sup>	500 mW
Input Voltages <sup>3</sup>	$\pm 18\text{ V}$
Output Short Circuit Duration	Indefinite
Storage Temperature Range	$-65^\circ\text{C to } +150^\circ\text{C}$
Operating Temperature Range	$0^\circ\text{C to } +70^\circ\text{C}$
Lead Temperature Range (Soldering 60 sec)	$+300^\circ\text{C}$
ESD Rating	1000 V

## NOTES

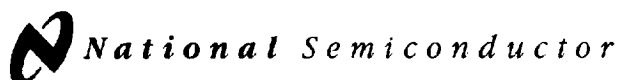
<sup>1</sup>Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied.

<sup>2</sup>8-Pin Plastic Package:  $\theta_{JA} = 165^\circ\text{C/W}$ ; 8-Pin Small Outline Package:  $\theta_{JA} = 155^\circ\text{C/W}$ .

<sup>3</sup>For supply voltages less than  $\pm 18\text{ V}$ , the absolute maximum input voltage is equal to the supply voltage.

## ORDERING GUIDE

Model	Description	Package Option
AD633JN	8-Pin Plastic DIP	N-8
AD633JR	8-Pin Plastic SOIC	R-8
AD633JR-REEL	8-Pin Plastic SOIC	R-8



December 1994

## LF155/LF156/LF157 Series Monolithic JFET Input Operational Amplifiers

### General Description

These are the first monolithic JFET input operational amplifiers to incorporate well matched, high voltage JFETs on the same chip with standard bipolar transistors (BI-FET™ Technology). These amplifiers feature low input bias and offset currents/low offset voltage and offset voltage drift, coupled with offset adjust which does not degrade drift or common-mode rejection. The devices are also designed for high slew rate, wide bandwidth, extremely fast settling time, low voltage and current noise and a low 1/f noise corner.

### Advantages

- Replace expensive hybrid and module FET op amps
- Rugged JFETs allow blow-out free handling compared with MOSFET input devices
- Excellent for low noise applications using either high or low source impedance—very low 1/f corner
- Offset adjust does not degrade drift or common-mode rejection as in most monolithic amplifiers
- New output stage allows use of large capacitive loads (5,000 pF) without stability problems
- Internal compensation and large differential input voltage capability

### Applications

- Precision high speed integrators
- Fast D/A and A/D converters
- High impedance buffers
- Wideband, low noise, low drift amplifiers
- Logarithmic amplifiers

- Photocell amplifiers
- Sample and Hold circuits

### Common Features

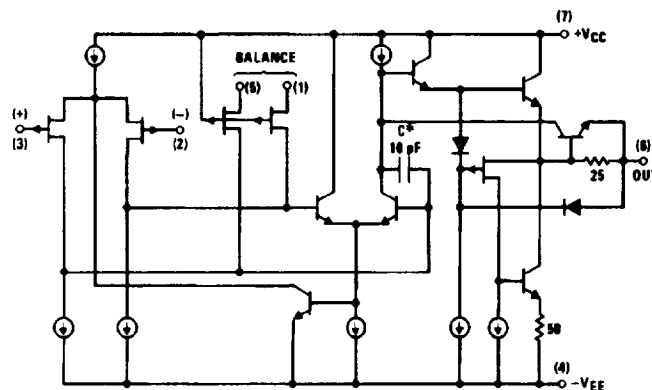
(LF155A, LF156A, LF157A)

- Low input bias current 30 pA
- Low Input Offset Current 3 pA
- High input impedance  $10^{12} \Omega$
- Low input offset voltage 1 mV
- Low input offset voltage temp. drift  $3 \mu V/^{\circ}C$
- Low input noise current  $0.01 \text{ pA}/\sqrt{\text{Hz}}$
- High common-mode rejection ratio 100 dB
- Large dc voltage gain 106 dB

### Uncommon Features

	LF155A	LF156A	LF157A ( $A_v=5$ )	Units
■ Extremely fast settling time to 0.01%	4	1.5	1.5	$\mu s$
■ Fast slew rate	5	12	50	$V/\mu s$
■ Wide gain bandwidth	2.5	5	20	MHz
■ Low input noise voltage	20	12	12	$nV/\sqrt{\text{Hz}}$

### Simplified Schematic



\*3 pF in LF157 series.

TL/H/5646-1

BI-FET™, BI-FET II™ are trademarks of National Semiconductor Corporation.

© 1995 National Semiconductor Corporation TL/H/5646

RRD-B30M115/Printed in U. S. A.

LF155/LF156/LF157 Series Monolithic JFET Input Operational Amplifiers

## Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/Distributors for availability and specifications.

(Note 8)

	LF155A/6A/7A	LF155/6/7	LF355B/6B/7B LF255/6/7	LF355/6/7 LF355A/6A/7A
Supply Voltage	$\pm 22\text{V}$	$\pm 22\text{V}$	$\pm 22\text{V}$	$\pm 18\text{V}$
Differential Input Voltage	$\pm 40\text{V}$	$\pm 40\text{V}$	$\pm 40\text{V}$	$\pm 30\text{V}$
Input Voltage Range (Note 2)	$\pm 20\text{V}$	$\pm 20\text{V}$	$\pm 20\text{V}$	$\pm 16\text{V}$
Output Short Circuit Duration	Continuous	Continuous	Continuous	Continuous
$T_{J\text{MAX}}$				
H-Package	150°C	150°C	115°C	115°C
N-Package			100°C	100°C
M-Package			100°C	100°C
Power Dissipation at $T_A = 25^\circ\text{C}$ (Notes 1 and 9)				
H-Package (Still Air)	560 mW	560 mW	400 mW	400 mW
H-Package (400 LF/Min Air Flow)	1200 mW	1200 mW	1000 mW	1000 mW
N-Package			670 mW	670 mW
M-Package			380 mW	380 mW
Thermal Resistance (Typical) $\theta_{JA}$				
H-Package (Still Air)	160°C/W	160°C/W	160°C/W	160°C/W
H-Package (400 LF/Min Air Flow)	65°C/W	65°C/W	65°C/W	65°C/W
N-Package			130°C/W	130°C/W
M-Package			195°C/W	195°C/W
(Typical) $\theta_{JC}$				
H-Package	23°C/W	23°C/W	23°C/W	23°C/W
Storage Temperature Range	-65°C to +150°C	-65°C to +150°C	-65°C to +150°C	-65°C to +150°C
Soldering Information (Lead Temp.)				
Metal Can Package				
Soldering (10 sec.)	300°C	300°C	300°C	300°C
Dual-In-Line Package				
Soldering (10 sec.)		260°C	260°C	260°C
Small Outline Package				
Vapor Phase (60 sec.)			215°C	215°C
Infrared (15 sec.)			220°C	220°C

See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.

ESD tolerance

(100 pF discharged through 1.5 k $\Omega$ )

1000V

1000V

1000V

1000V

## DC Electrical Characteristics (Note 3) $T_A = T_J = 25^\circ\text{C}$

Symbol	Parameter	Conditions	LF155A/6A/7A			LF355A/6A/7A			Units
			Min	Typ	Max	Min	Typ	Max	
$V_{OS}$	Input Offset Voltage	$R_S = 50\Omega$ , $T_A = 25^\circ\text{C}$ Over Temperature		1	2 2.5		1	2 2.3	mV mV
$\Delta V_{OS}/\Delta T$	Average TC of Input Offset Voltage	$R_S = 50\Omega$		3	5		3	5	$\mu\text{V}/^\circ\text{C}$
$\Delta TC/\Delta V_{OS}$	Change in Average TC with $V_{OS}$ Adjust	$R_S = 50\Omega$ , (Note 4)		0.5			0.5		$\mu\text{V}/^\circ\text{C}$ per mV
$I_{OS}$	Input Offset Current	$T_J = 25^\circ\text{C}$ , (Notes 3, 5) $T_J \leq T_{HIGH}$		3	10 10		3	10 1	pA nA
$I_B$	Input Bias Current	$T_J = 25^\circ\text{C}$ , (Notes 3, 5) $T_J \leq T_{HIGH}$		30	50 25		30	50 5	pA nA
$R_{IN}$	Input Resistance	$T_J = 25^\circ\text{C}$		10 <sup>12</sup>			10 <sup>12</sup>		$\Omega$
$A_{VOL}$	Large Signal Voltage Gain	$V_S = \pm 15\text{V}$ , $T_A = 25^\circ\text{C}$ $V_O = \pm 10\text{V}$ , $R_L = 2\text{k}$ Over Temperature	50 25	200		50 25	200		V/mV V/mV
$V_O$	Output Voltage Swing	$V_S = +15\text{V}$ , $R_L = 10\text{k}$ $V_S = \pm 15\text{V}$ , $R_L = 2\text{k}$	$\pm 12$ $\pm 10$	$\pm 13$ $\pm 12$		$\pm 12$ $\pm 10$	$\pm 13$ $\pm 12$		V V

## Absolute Maximum Ratings

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/Distributors for availability and specifications.

(Note 8)

	LF155A/6A/7A	LF155/6/7	LF355B/6B/7B LF255/6/7	LF355/6/7 LF355A/6A/7A
Supply Voltage	± 22V	± 22V	± 22V	± 18V
Differential Input Voltage	± 40V	± 40V	± 40V	± 30V
Input Voltage Range (Note 2)	± 20V	± 20V	± 20V	± 16V
Output Short Circuit Duration	Continuous	Continuous	Continuous	Continuous
$T_{JMAX}$				
H-Package	150°C	150°C	115°C	115°C
N-Package			100°C	100°C
M-Package			100°C	100°C
Power Dissipation at $T_A = 25^\circ\text{C}$ (Notes 1 and 9)				
H-Package (Still Air)	560 mW	560 mW	400 mW	400 mW
H-Package (400 LF/Min Air Flow)	1200 mW	1200 mW	1000 mW	1000 mW
N-Package			670 mW	670 mW
M-Package			380 mW	380 mW
Thermal Resistance (Typical) $\theta_{JA}$				
H-Package (Still Air)	160°C/W	160°C/W	160°C/W	160°C/W
H-Package (400 LF/Min Air Flow)	65°C/W	65°C/W	65°C/W	65°C/W
N-Package			130°C/W	130°C/W
M-Package			195°C/W	195°C/W
(Typical) $\theta_{JC}$				
H-Package	23°C/W	23°C/W	23°C/W	23°C/W
Storage Temperature Range	-65°C to +150°C	-65°C to +150°C	-65°C to +150°C	-65°C to +150°C
Soldering Information (Lead Temp.)				
Metal Can Package				
Soldering (10 sec.)	300°C	300°C	300°C	300°C
Dual-In-Line Package				
Soldering (10 sec.)		260°C	260°C	260°C
Small Outline Package				
Vapor Phase (60 sec.)			215°C	215°C
Infrared (15 sec.)			220°C	220°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.				
ESD tolerance				
(100 pF discharged through 1.5 k $\Omega$ )	1000V	1000V	1000V	1000V

## DC Electrical Characteristics (Note 3) $T_A = T_j = 25^\circ\text{C}$

Symbol	Parameter	Conditions	LF155A/6A/7A			LF355A/6A/7A			Units
			Min	Typ	Max	Min	Typ	Max	
$V_{OS}$	Input Offset Voltage	$R_S = 50\Omega$ , $T_A = 25^\circ\text{C}$ Over Temperature		1	2 2.5		1	2 2.3	mV mV
$\Delta V_{OS}/\Delta T$	Average TC of Input Offset Voltage	$R_S = 50\Omega$		3	5		3	5	$\mu\text{V}/^\circ\text{C}$
$\Delta TC/\Delta V_{OS}$	Change in Average TC with $V_{OS}$ Adjust	$R_S = 50\Omega$ , (Note 4)		0.5			0.5		$\mu\text{V}/^\circ\text{C}$ per mV
$I_{OS}$	Input Offset Current	$T_j = 25^\circ\text{C}$ , (Notes 3, 5) $T_j \leq T_{HIGH}$		3	10 10		3	10 1	pA nA
$I_B$	Input Bias Current	$T_j = 25^\circ\text{C}$ , (Notes 3, 5) $T_j \leq T_{HIGH}$		30	50 25		30	50 5	pA nA
$R_{IN}$	Input Resistance	$T_j = 25^\circ\text{C}$		10 <sup>12</sup>			10 <sup>12</sup>		$\Omega$
$A_{VOL}$	Large Signal Voltage Gain	$V_S = \pm 15\text{V}$ , $T_A = 25^\circ\text{C}$ $V_O = \pm 10\text{V}$ , $R_L = 2\text{k}$ Over Temperature	50 25	200		50 25	200		V/mV V/mV
$V_O$	Output Voltage Swing	$V_S = \pm 15\text{V}$ , $R_L = 10\text{k}$ $V_S = \pm 15\text{V}$ , $R_L = 2\text{k}$	± 12 ± 10	± 13 ± 12		± 12 ± 10	± 13 ± 12		V V

**DC Electrical Characteristics**  $T_A = T_J = 25^\circ\text{C}$ ,  $V_S = \pm 15\text{V}$ 

Parameter	LF155A/155, LF255, LF355A/355B		LF355		LF156A/156, LF256/356B		LF356A/356		LF157A/157 LF257/357B		LF357A/357		Units
	Typ	Max	Typ	Max	Typ	Max	Typ	Max	Typ	Max	Typ	Max	
Supply Current	2	4	2	4	5	7	5	10	5	7	5	10	mA

**AC Electrical Characteristics**  $T_A = T_J = 25^\circ\text{C}$ ,  $V_S = +15\text{V}$ 

Symbol	Parameter	Conditions	LF155/255/ 355/355B	LF156/256, LF356B	LF156/256/ 356/356B	LF157/257, LF357B	LF157/257/ 357/357B	Units
			Typ	Min	Typ	Min	Typ	
SR	Slew Rate	LF155/6: $A_V = 1$ , LF157: $A_V = 5$	5	7.5	12	30	50	V/ $\mu\text{s}$ V/ $\mu\text{s}$
GBW	Gain Bandwidth Product		2.5		5		20	MHz
$t_s$	Settling Time to 0.01%	(Note 7)	4		1.5		1.5	$\mu\text{s}$
$e_n$	Equivalent Input Noise Voltage	$R_S = 100\Omega$ $f = 100\text{ Hz}$ $f = 1000\text{ Hz}$	25 20		15 12		15 12	nV/ $\sqrt{\text{Hz}}$ nV/ $\sqrt{\text{Hz}}$
$i_n$	Equivalent Input Current Noise	$f = 100\text{ Hz}$ $f = 1000\text{ Hz}$	0.01 0.01		0.01 0.01		0.01 0.01	pA/ $\sqrt{\text{Hz}}$ pA/ $\sqrt{\text{Hz}}$
$C_{IN}$	Input Capacitance		3		3		3	pF

**Notes for Electrical Characteristics**

**Note 1:** The maximum power dissipation for these devices must be derated at elevated temperatures and is dictated by  $T_{JMAX}$ ,  $\theta_{JA}$ , and the ambient temperature,  $T_A$ . The maximum available power dissipation at any temperature is  $P_d = (T_{JMAX} - T_A)/\theta_{JA}$  or the  $25^\circ\text{C}$   $P_{dMAX}$ , whichever is less.

**Note 2:** Unless otherwise specified the absolute maximum negative input voltage is equal to the negative power supply voltage.

**Note 3:** Unless otherwise stated, these test conditions apply:

	LF155A/6A/7A LF155/6/7	LF255/6/7	LF355A/6A/7A	LF355B/6B/7B	LF355/6/7
Supply Voltage, $V_S$	$\pm 15\text{V} \leq V_S \leq \pm 20\text{V}$	$\pm 15\text{V} \leq V_S \leq \pm 20\text{V}$	$\pm 15\text{V} \leq V_S \leq \pm 18\text{V}$	$\pm 15\text{V} \leq V_S \leq \pm 20\text{V}$	$V_S = \pm 15\text{V}$
$T_A$	$-55^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$	$-25^\circ\text{C} \leq T_A \leq +85^\circ\text{C}$	$0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$	$0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$	$0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$
$T_{HIGH}$	$+125^\circ\text{C}$	$+85^\circ\text{C}$	$+70^\circ\text{C}$	$+70^\circ\text{C}$	$+70^\circ\text{C}$

and  $V_{OS}$ ,  $I_B$  and  $I_{OS}$  are measured at  $V_{CM} = 0$ .

**Note 4:** The Temperature Coefficient of the adjusted input offset voltage changes only a small amount ( $0.5\mu\text{V}/^\circ\text{C}$  typically) for each mV of adjustment from its original unadjusted value. Common-mode rejection and open loop voltage gain are also unaffected by offset adjustment.

**Note 5:** The input bias currents are junction leakage currents which approximately double for every  $10^\circ\text{C}$  increase in the junction temperature,  $T_J$ . Due to limited production test time, the input bias currents measured are correlated to junction temperature. In normal operation the junction temperature rises above the ambient temperature as a result of internal power dissipation,  $P_d$ .  $T_J = T_A + \theta_{JA} P_d$  where  $\theta_{JA}$  is the thermal resistance from junction to ambient. Use of a heat sink is recommended if input bias current is to be kept to a minimum.

**Note 6:** Supply Voltage Rejection is measured for both supply magnitudes increasing or decreasing simultaneously, in accordance with common practice.

**Note 7:** Settling time is defined here, for a unity gain inverter connection using  $2\text{ k}\Omega$  resistors for the LF155/6. It is the time required for the error voltage (the voltage at the inverting input pin on the amplifier) to settle to within 0.01% of its final value from the time a  $10\text{V}$  step input is applied to the inverter. For the LF157,  $A_V = -5$ , the feedback resistor from output to input is  $2\text{ k}\Omega$  and the output step is  $10\text{V}$  (See Settling Time Test Circuit).

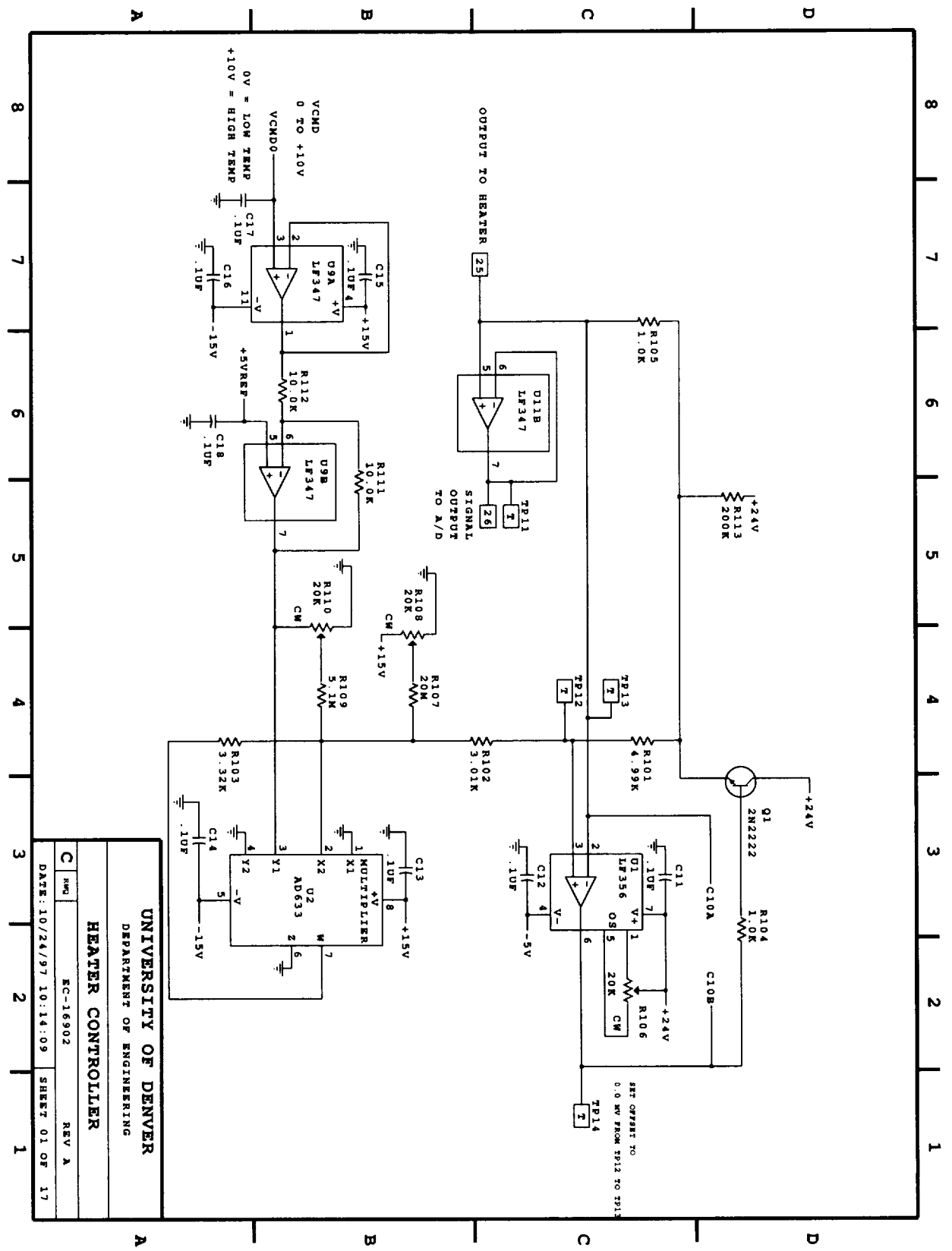
**Note 8:** Refer to RETS155AX for LF155A, RETS155X for LF155, RETS156AX for LF156A, RETS156X for LF156, RETS157A for LF157A and RETS157X for LF157 military specifications.

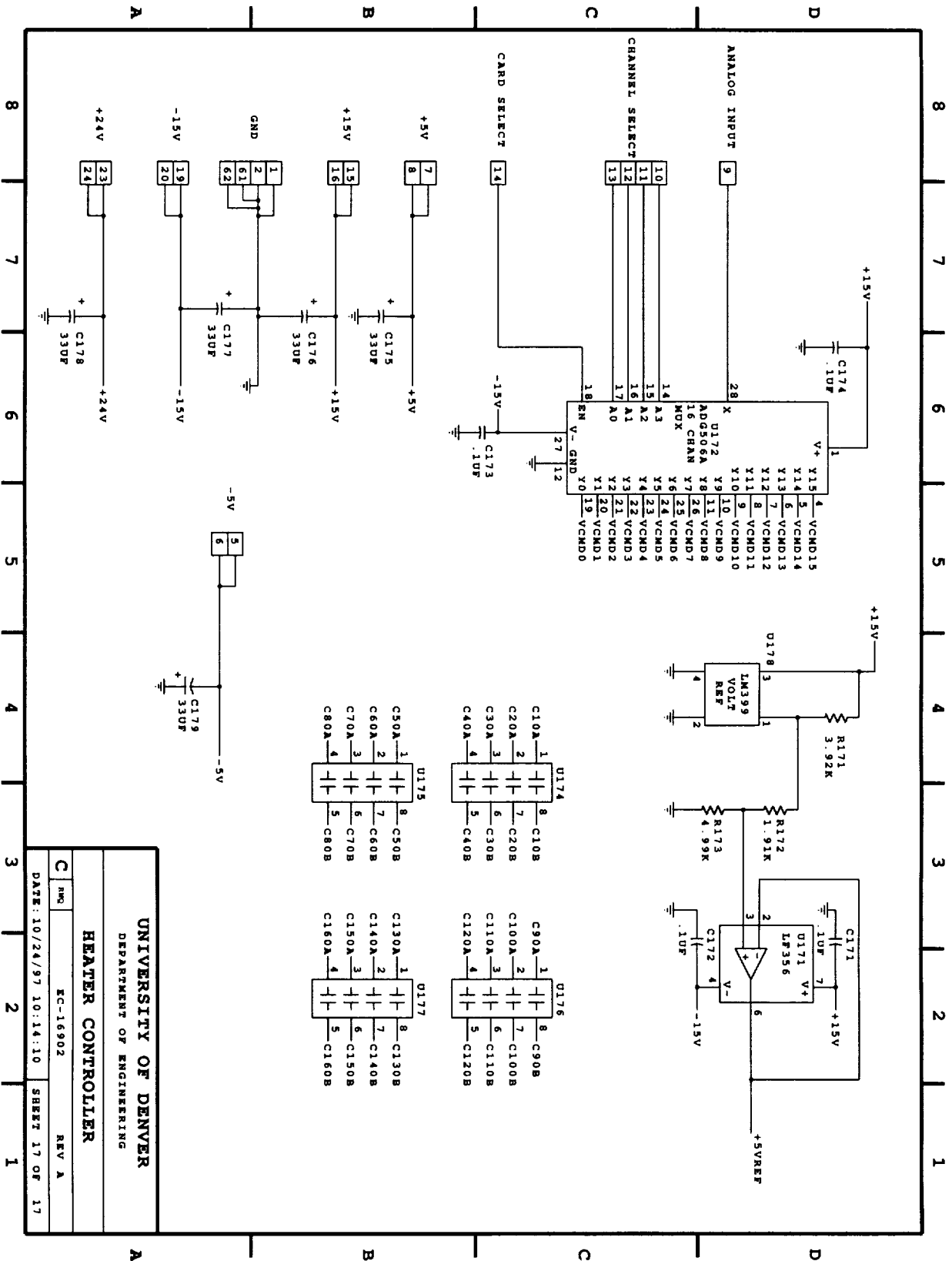
**Note 9:** Max. Power Dissipation is defined by the package characteristics. Operating the part near the Max. Power Dissipation may cause the part to operate outside guaranteed limits.

## **APPENDIX F: CONTROL SYSTEM SCHEMATICS**

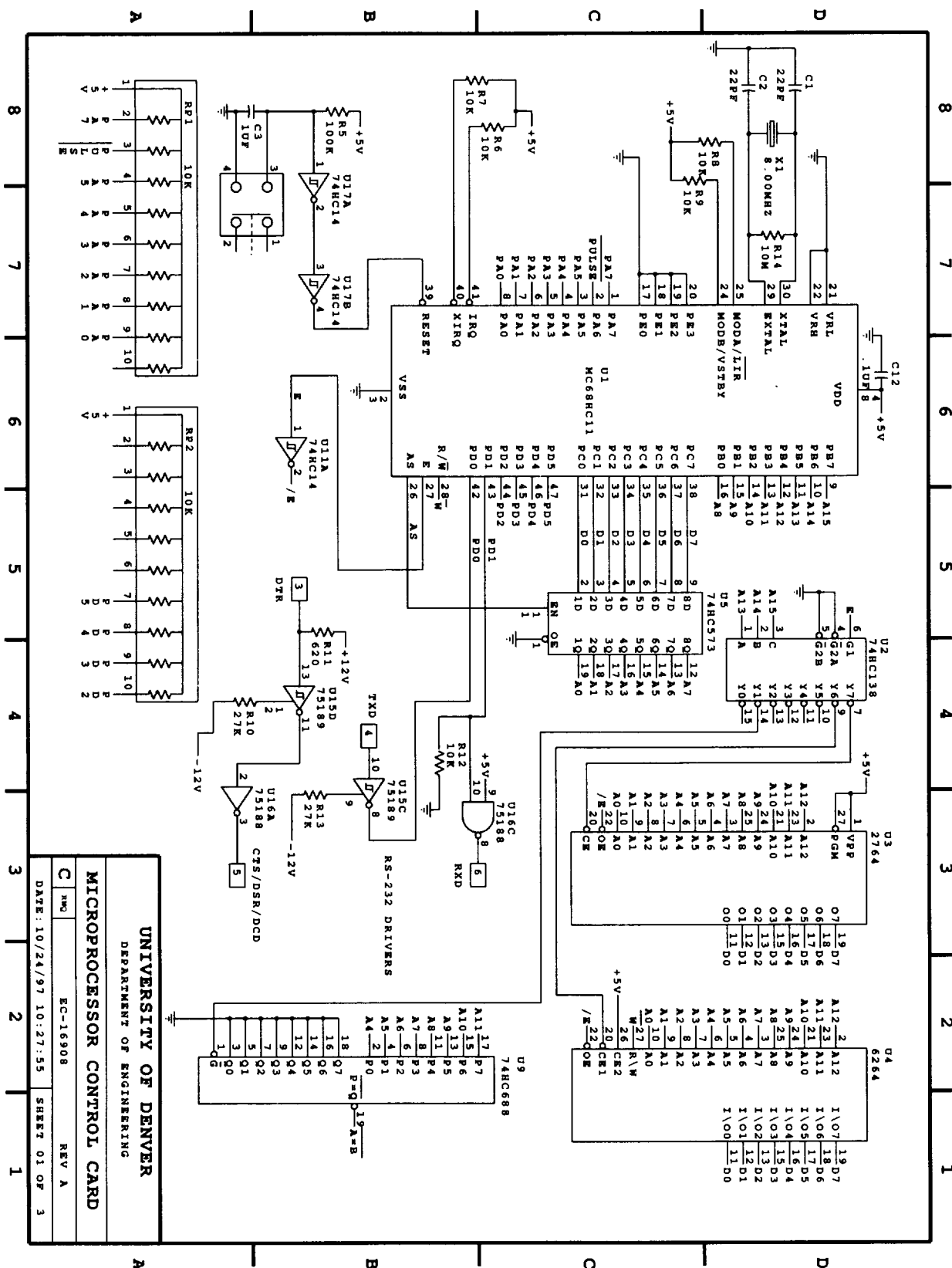
This section contains the complete schematics and part lists for the control system, including the feedback control board in section F.1. , the microprocessor control board in Section F.2. , the custom high-speed A/D board in Section F.3. , and the motherboard in Section F.4. . Only the first feedback board schematic is shown, because the other 15 feedback circuits on each board are identical.

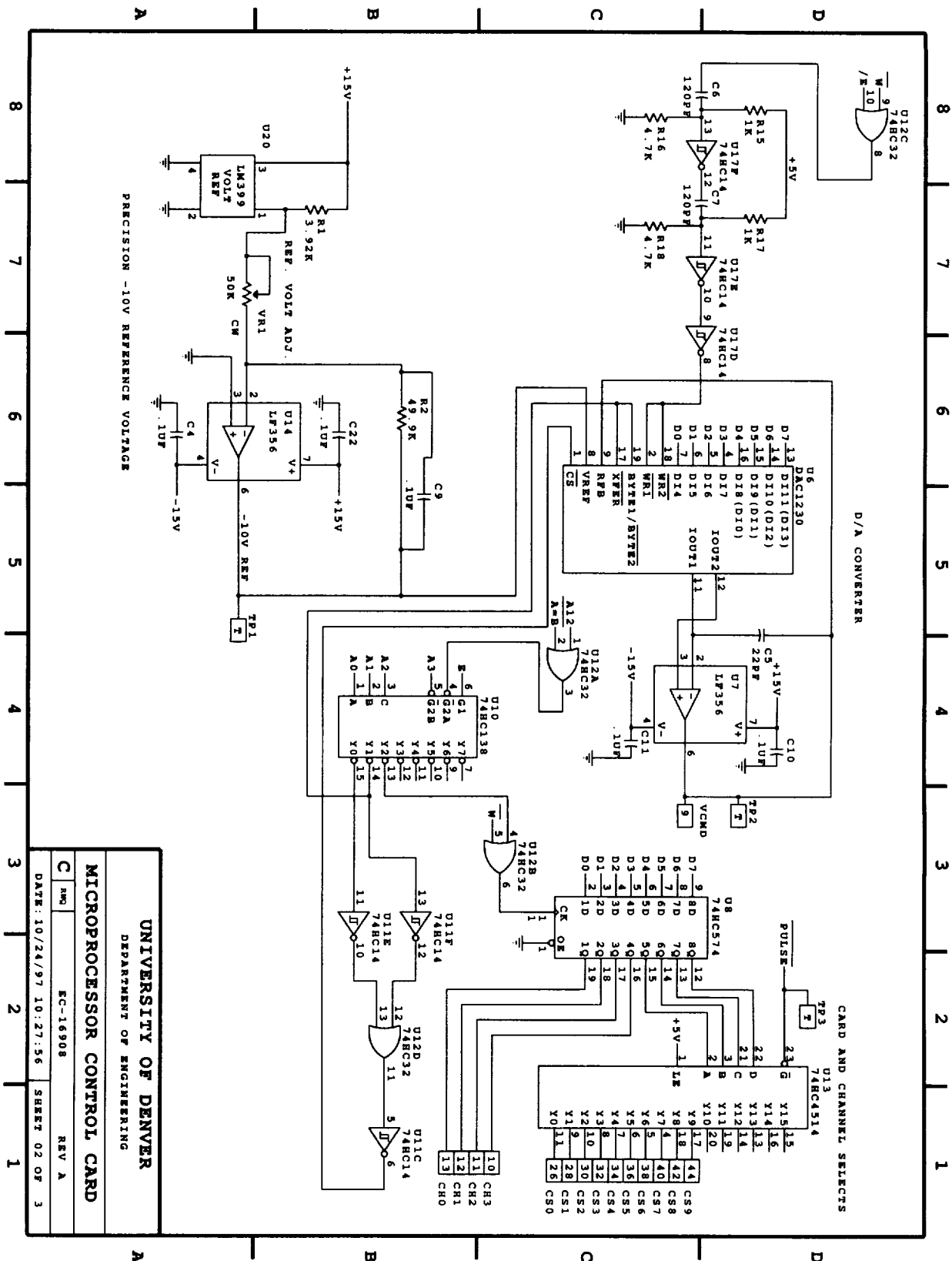
F.1. FEEDBACK CONTROL BOARD SCHEMATIC

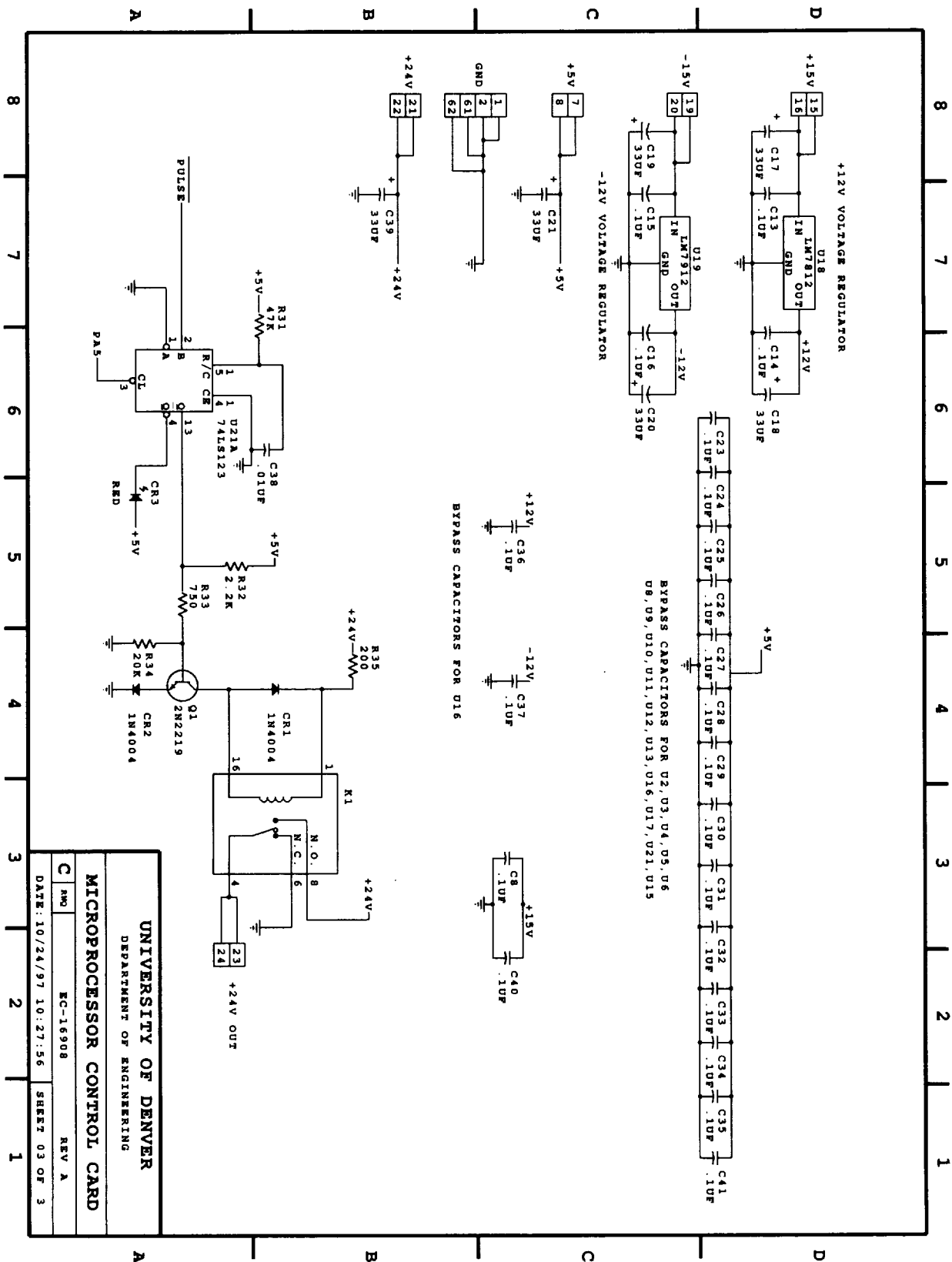




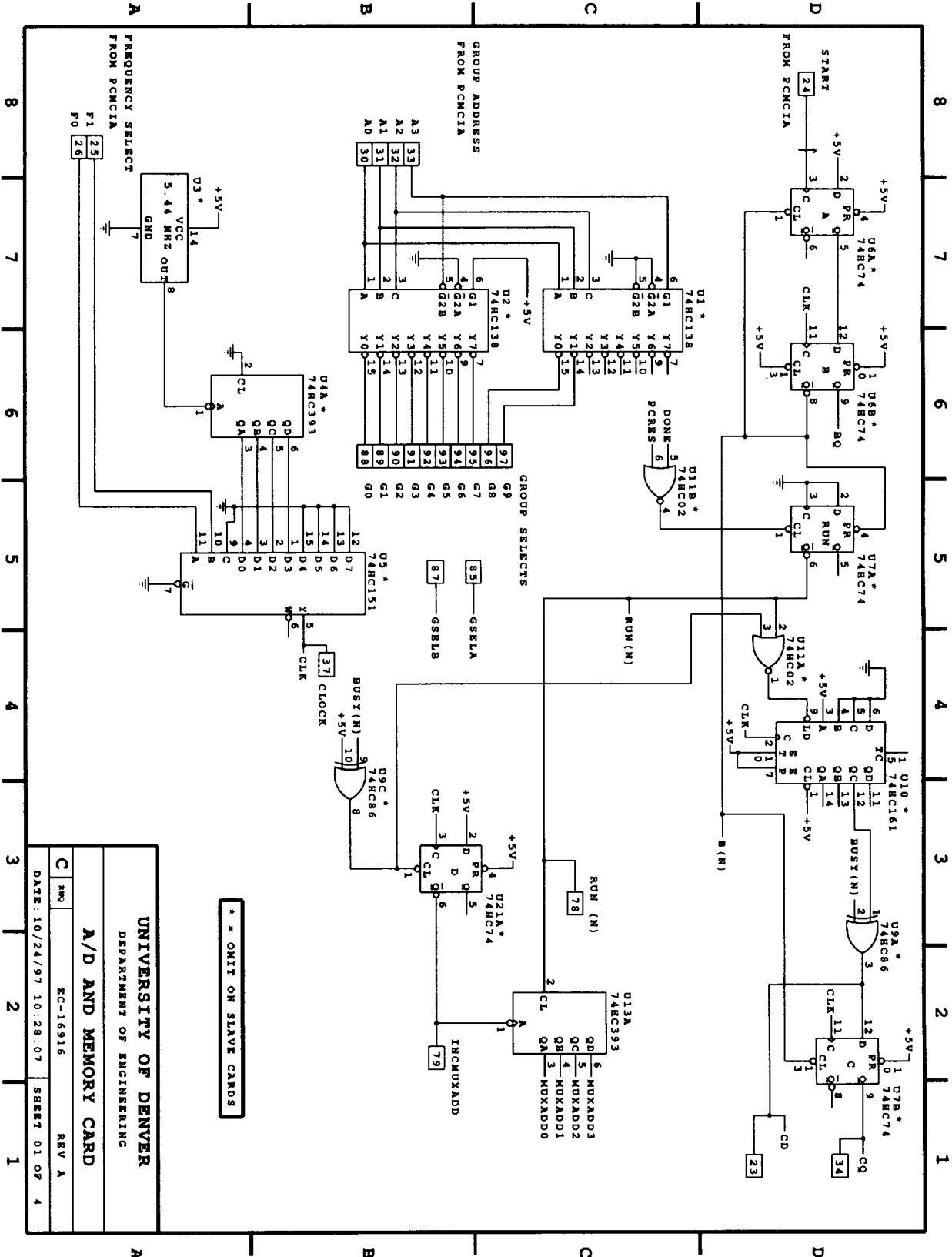


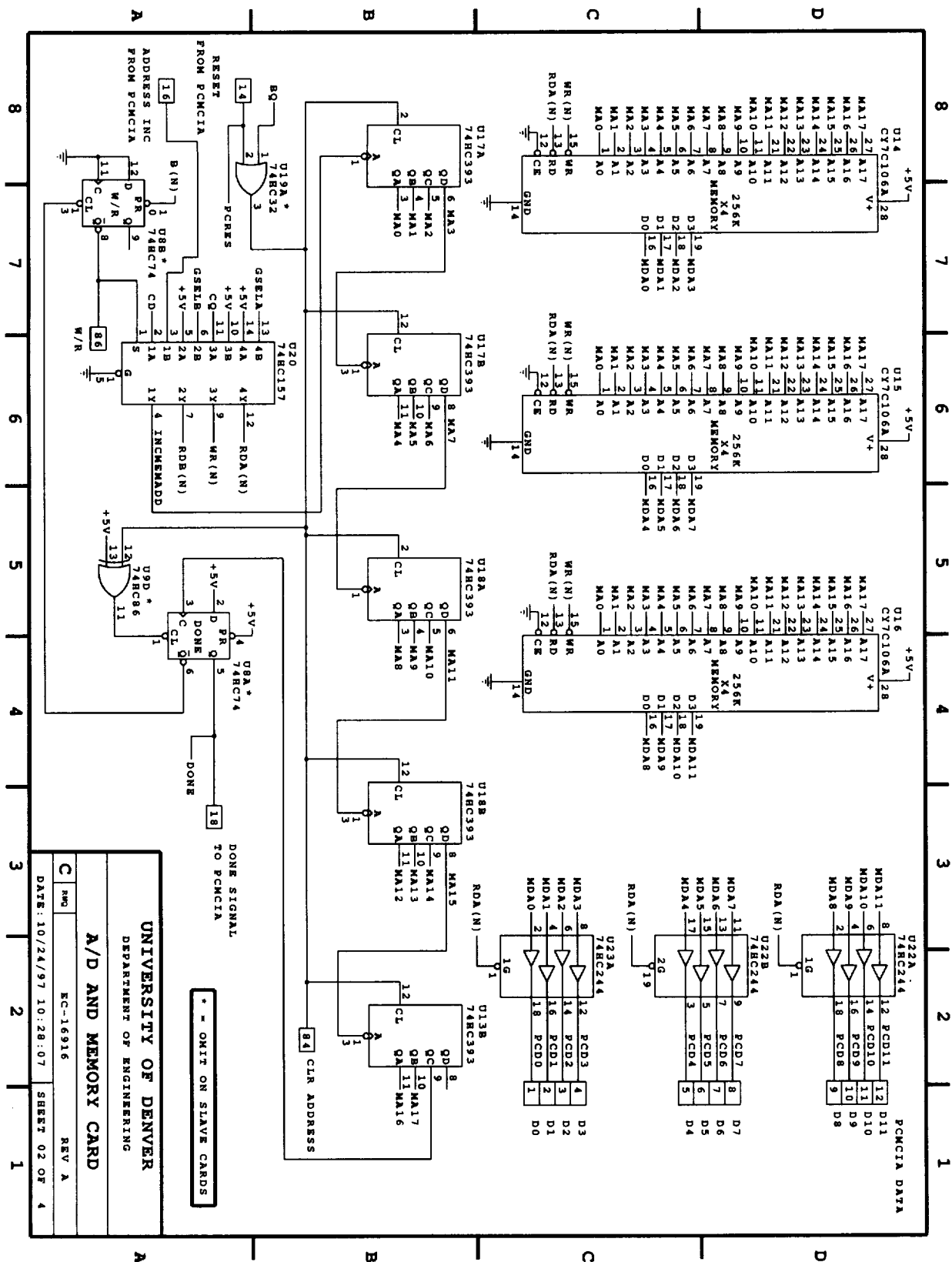




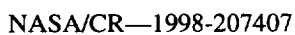


## F.3. CUSTOM HIGH-SPEED A/D BOARD

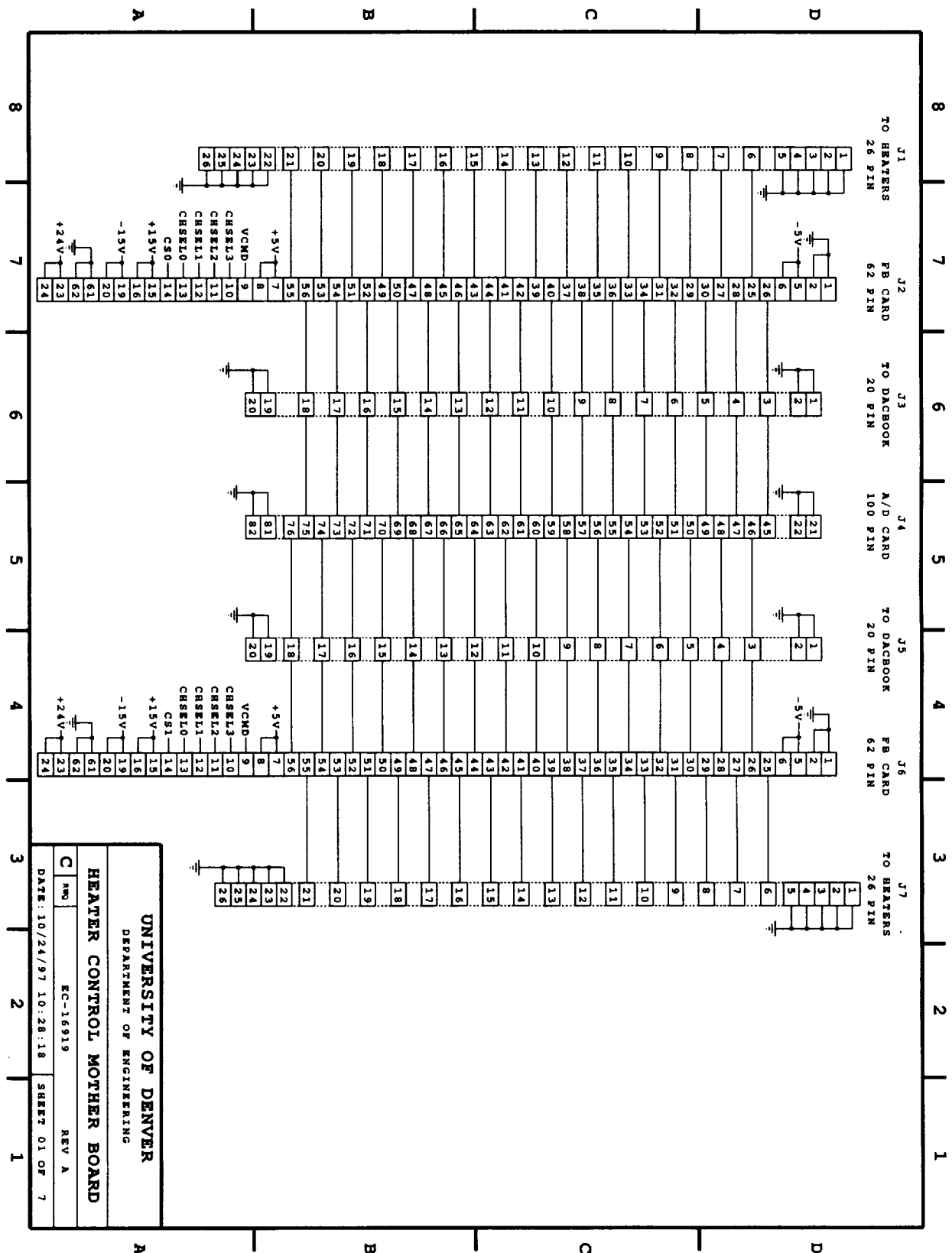




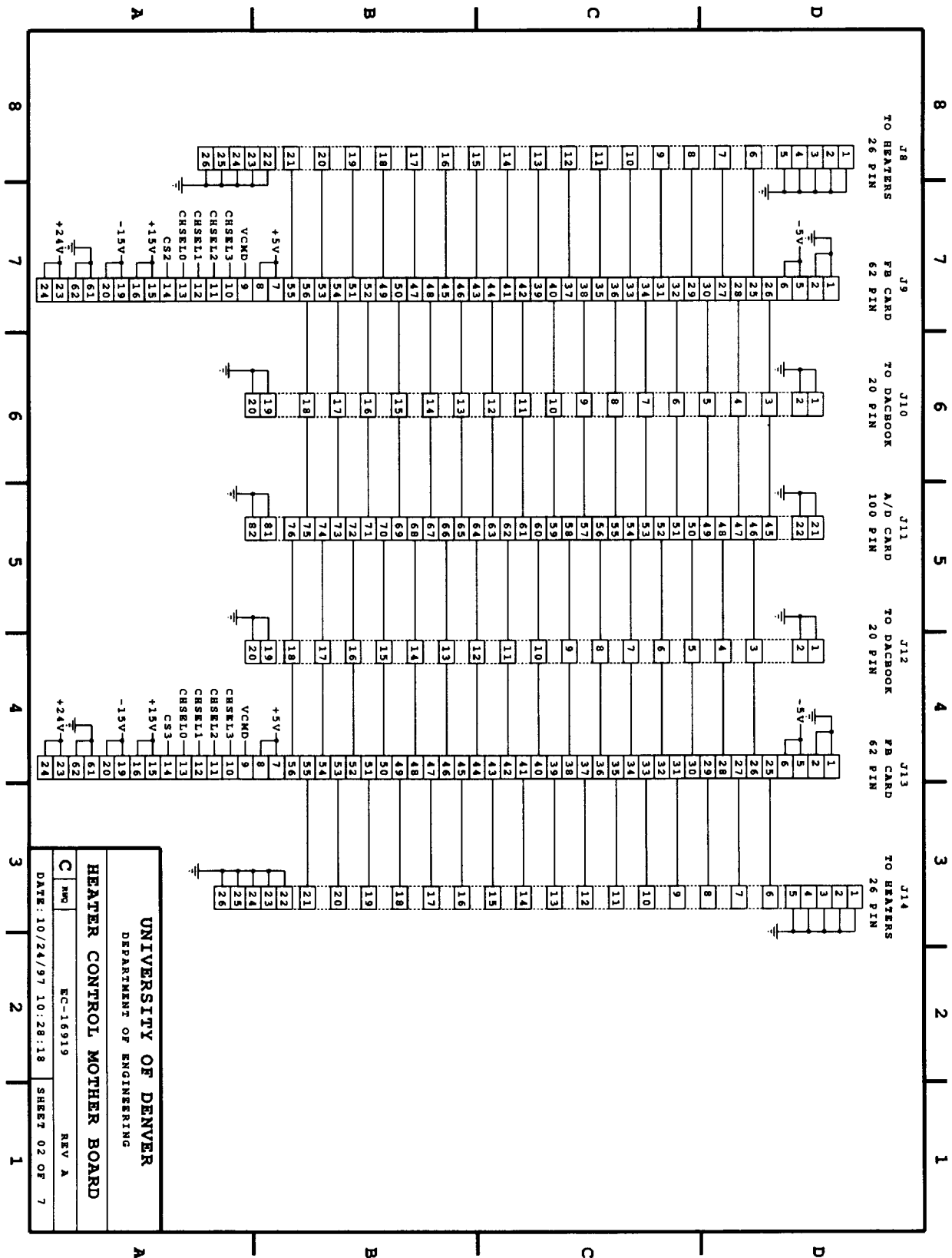


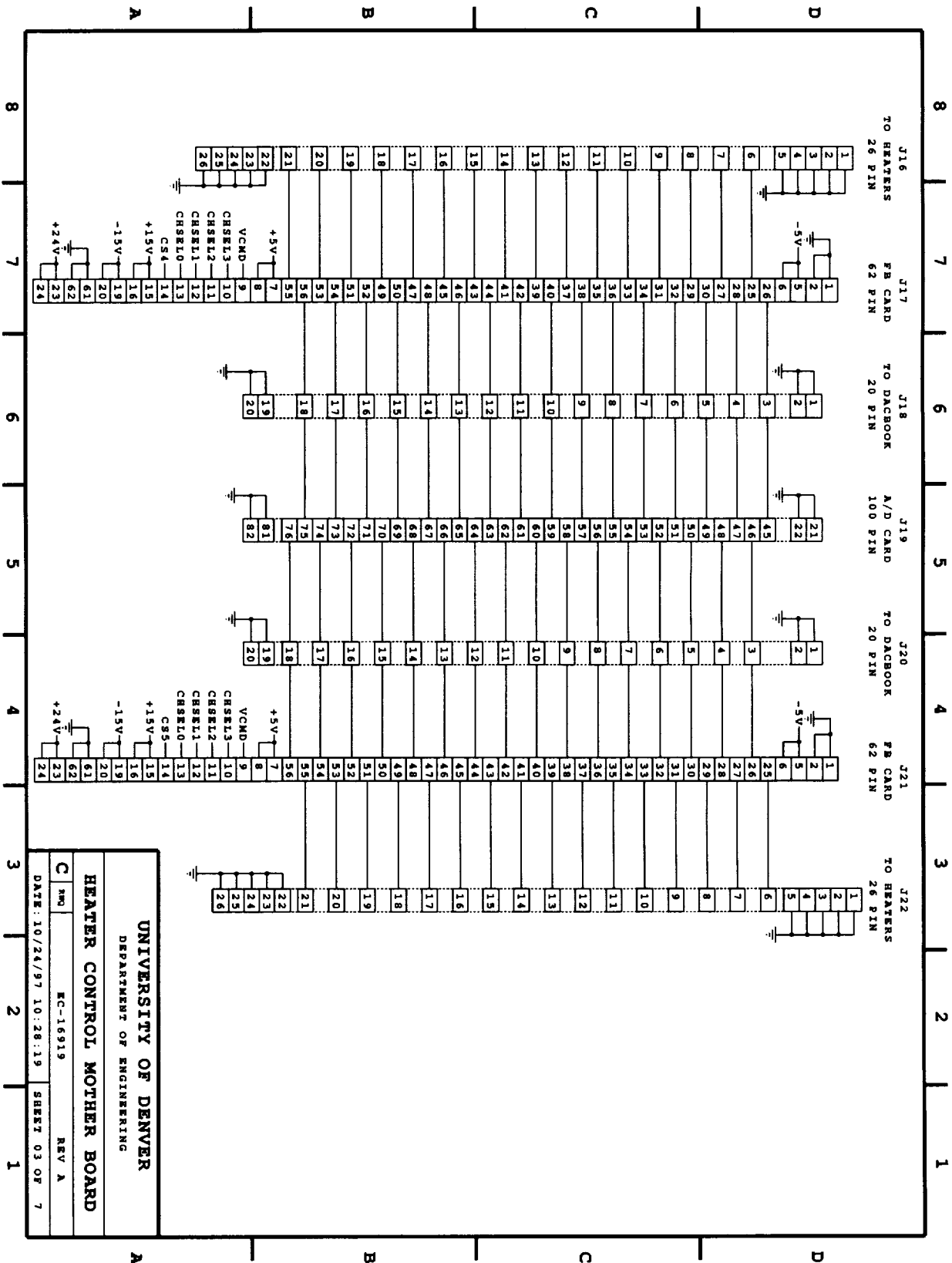


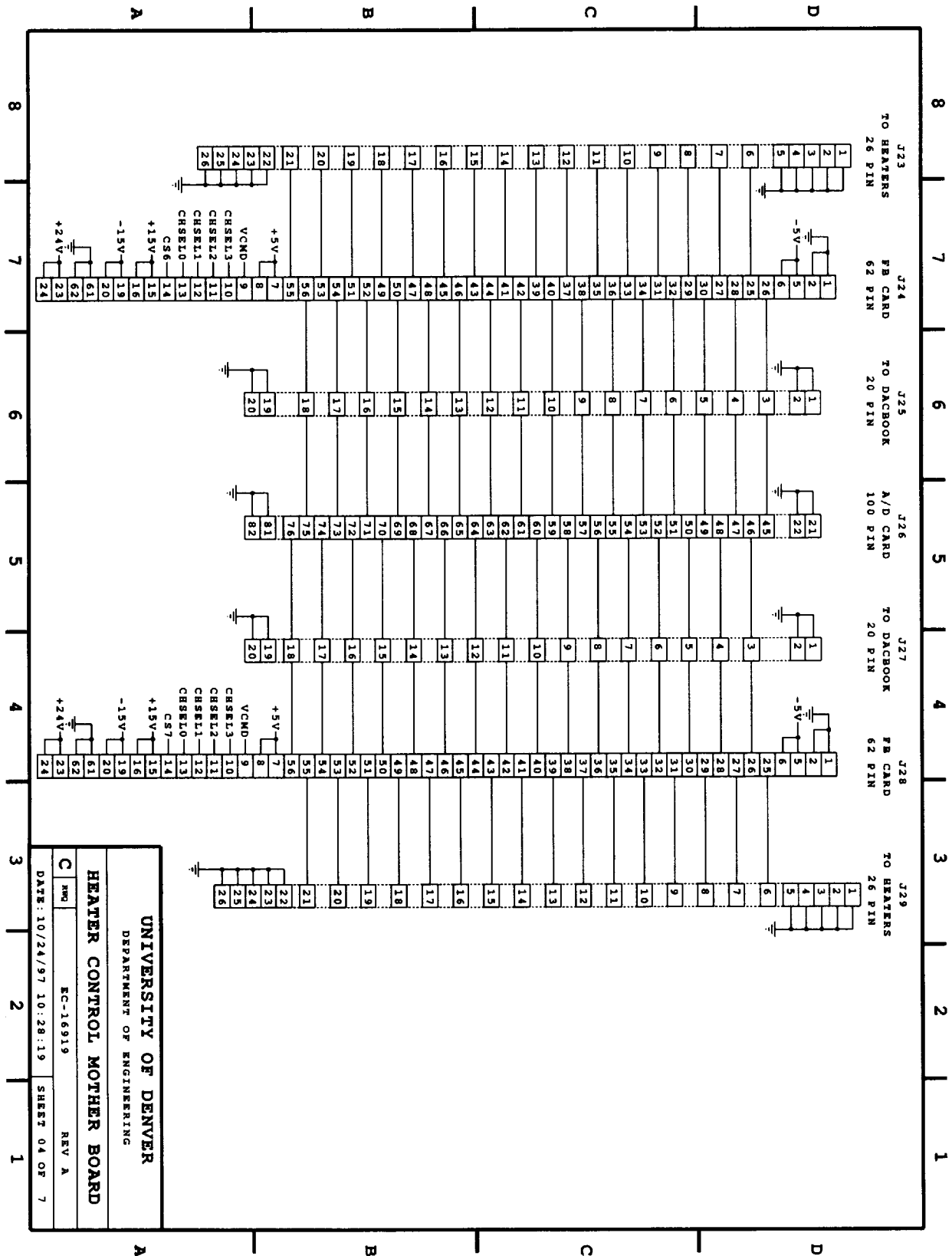
## F.4. MOTHERBOARD SCHEMATIC

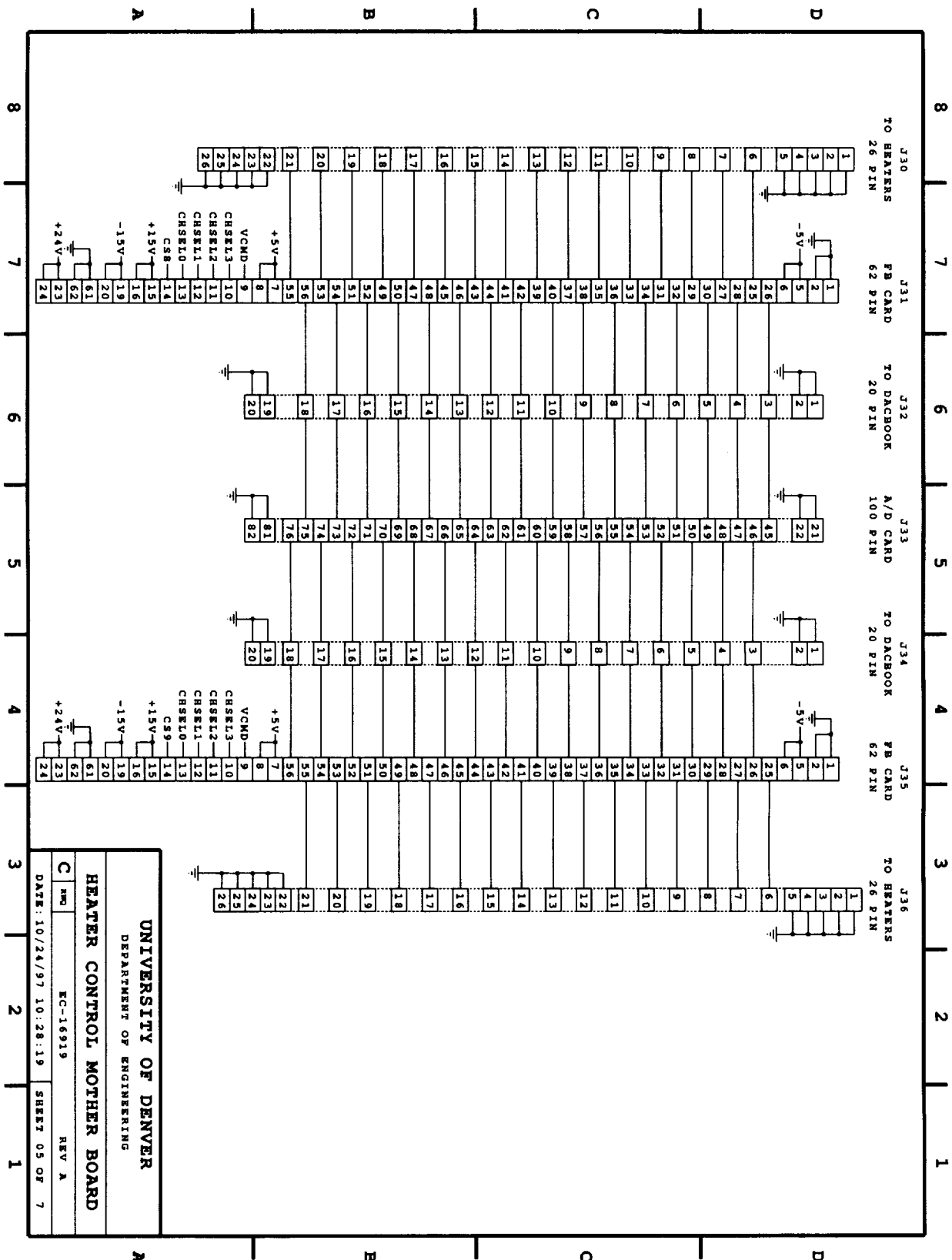


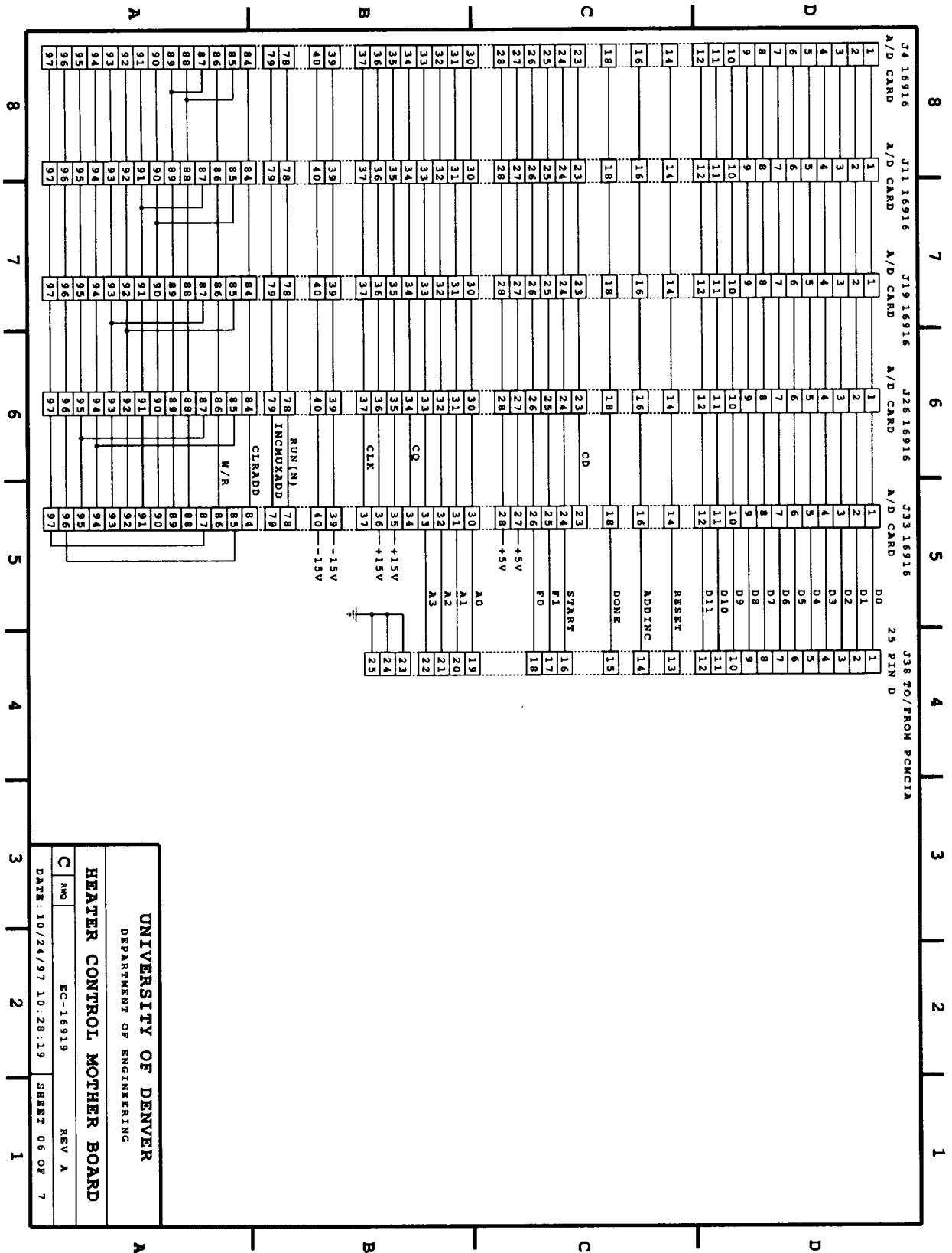


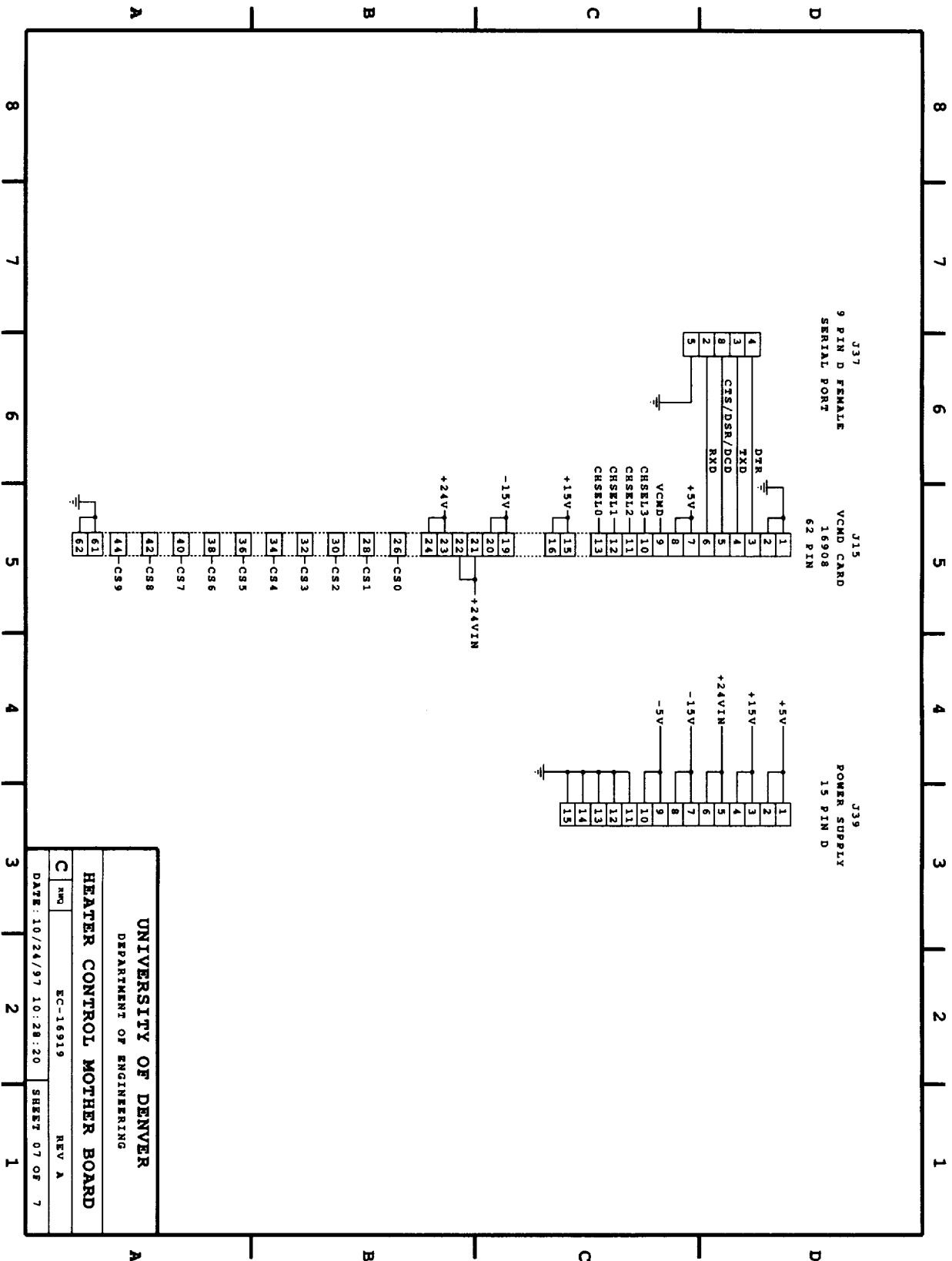












## APPENDIX G: SOURCE CODE LISTINGS FOR VISUAL BASIC PROGRAMS

The following chapters contain the source code listings for the programs that are described in Appendix B.

### G.1. LISTING OF "CONTROL.VBP"

```
Module=modADC; ..\TRIM\MODADC.BAS
Module=modDaqBook; ..\DAQBOOK.BAS
Module=modShared; MODSHARE.BAS
Module=modCBW; ..\..\..\CB\VBWIN\CBW.BAS
Module=modConvert; MODCONVE.BAS
Form=FRMSETUP.FRM
Form=..\TRIM\FRMCONFI.FRM
Form=FRMVIEWS.FRM
Form=..\TRIM\FRMDAC.FRM
Form=FRMADC.FRM
Form=FRMCONVE.FRM
Form=FRMAUTOM.FRM
Form=FRMCALFI.FRM
Form=FRMAUTO.FRM
Form=FRMBATCH.FRM
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMDLG16.OCX
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL16.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST16.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID16.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#1.0#0#C:\WINDOWS\SYSTEM\OC25.DLL#Standard OLE
Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#2.5#0#C:\WINDOWS\SYSTEM\DAO2516.DLL#Microsoft
DAO 2.5 Object Library
Object={648A5603-2C6E-101B-82B6-000000000014}#1.0#0; MSCOMM16.OCX
Object={A8B3B723-0B5A-101B-B22E-00AA0037B2FC}#1.0#0; GRID16.OCX
ProjWinSize=55,360,251,397
ProjWinShow=2
IconForm="frmMain"
HelpFile=""
ExeName="CONTROL.EXE"
Path="F:\BIN"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="University of Denver"
```

#### G.1.1. Listing of MODADC.BAS

```
Attribute VB_Name = "modADC"
Option Explicit
'*****
' * This module contains code for declaring the A/D
' * data types and for handling the low-level A/D
' * functions
'*****
```

```

*****
' * the ADC type is the data type that will be used
' * to interface with the A/D card. I'm trying to
' * design it so that the software won't have to know
' * anything more about the hardware than what's in
' * the ADC members, so that all the low-level
' * hardware details can be isolated in one module -
' * that is, this module.
*****

Public Type ADCTYPE
' * I don't anticipate needing different gains for
' * different channels, so I'm just using one gain
' * for all channels
Gain As Integer
' * Sampling rate in samples/second, for all channels
Freq As Integer
' * Channels is TRUE if active, FALSE if inactive
Channels(160) As Boolean
' * How many samples to take
numscans As Integer
' * How to trigger - Hardware, software, etc.
trigger As Integer
*****
' * I am going to map all data into a 2-D array,
' * even if the input routines read the data as
' * 1-D.
' *
' * This buffer will not be used if writing data
' * to disk.
' *
' * First index is the next highest channel read,
' * second index, is the data point.
*****

Buffer(96, 40) As Double
' * File name that data will be saved to when
' * writing directly to disk
filename As String
FileNumber As Integer
HardwareType As Integer
End Type
' * Specifies whether we are using Daqbook or Custom A/D
Public ADCHardwareType As Integer
' * Trigger source Constants
' * This specifies a software trigger
Public Const MANUALTRIG = 0
' * List of gains for the drop-down list-boxes
Public GainList As ListType
' * List of trigger choices for drop-down list-boxes
Public TriggerList As ListType
' * General ADC setup variable
Public ADC As ADCTYPE
' * to signal when ADC has been initialized
Public ADCInitialized As Boolean
Public Const ADCLASTCHANNEL = 95
Public Const ADCNOHARDWARE = True
Public Const ADCNONE = 0
Public Const ADCDAQBOOK = 1
Public Const ADCCUSTOM = 2

*****
' * The following are declarations pulled out of
' * dbk2disk.frm, from the daqbook software.
*****

' * This constant determines the number of elements in the
' * array that spools data to disk
Const ARRAYSIZE = 16000
Dim DataArray(1 To ARRAYSIZE) As Integer

```



```

Dim ScanSize As Integer
Dim MaxBufIndex As Integer
Dim ReadIndex As Integer
Dim ScansProcessed As Long
Dim FHandle As Integer
Dim er As Integer
Dim FileNumber As Integer

'* Declare the fast, low-level disk I/O function for getting data to
'* disk without the buffer overflowing.
Private Declare Function fWrite Lib "kernel" Alias "_lwrite" (ByVal hFile As Integer, lpBuff As
Any, ByVal wBytes As Integer) As Integer

'* ADCOffset is the number that is added to the physical channel number
'* that is sampled, but not to the Vcmd heater number that is output or the
'* heater number reference that is saved to disk.

'* EXAMPLE: I want to calibrate heaters 16-31 (16 heaters) using
'* the daqbook base unit (channels 0-15). I use an offset of -16. That
'* way, the outputs a command voltage to heater 16, but it will sample the
'* output voltage with channel 16+ADCOffset or 16+(-16)=0 -> channel 0
'* of the daqbook.

'* EXAMPLE: Use an offset of +16 for the expansion module, since expansion
'* module channel 0 is actually referenced as channel 16.

Public ADCOffset As Integer

'* Variables and constants for the custom A/D system

Const ADFREQ10000 = 0
Const ADFREQ5000 = 1
Const ADFREQ2500 = 2
Const ADFREQ1250 = 3
Const ADCBOARDNUM = 0

'* variable true if group contains channels to sample,
'* false if it can be skipped
Dim ADCGroups(10) As Integer
Dim ADCFirstGroup As Integer
Dim ADCLastGroup As Integer
Dim ADCBuffer(16000, 96) As Integer
Dim ADCFreq As Integer
Dim ADCGroupNum As Integer
Public Type ADCListType
    Num As Integer
    text(5) As String
    Value(5) As Integer
End Type
Public ADCList As ADCListType
'* parallel port protocol
Public ADCPPP As Integer
'* Extended Parallel Port on TI computer
Public Const ADCEPP = DaqProtocolSMC666
'* 8-bit protocol
Public Const ADC8BIT = DaqProtocol8

Public Sub ADC2Disk()

    Dim TriggerConst As Integer
    Dim chans(ADCLASTCHANNEL) As Integer
    Dim gains(ADCLASTCHANNEL) As Integer
    Dim stChan As Integer
    Dim endChan As Integer
    '* I already defined i. TDR
    Dim i As Integer
    Dim Cnt As Integer
    Dim Freq As Single
    Dim MaxScansInBuf As Integer
    Select Case ADCHardwareType
        Case ADCDAQBOOK

```

```

ScansProcessed = 0
ReadIndex = 1
' * Returns the operating system file handle
FHandle = FileAttr(ADC.FileName, 2)

' * I'm getting rid of these lines because I
' * want to be able to specify any combination
' * of channels. TDR

'stChan = Val(StartChannelTextBox.Text)
'endChan = Val(EndChannelTextBox.Text)

' * disable channel tagging
er = VBdaqAdcSetTag(0)

'Configure the channels with their gains.
'Each channel can have a different gain, but for simplicity's
'sake, I've made them all the same.
'cnt = 0
'For i = stChan To endChan
'    chans(cnt) = i
'    gains(cnt) = GainCombo.ListIndex
'    cnt = cnt + 1
'Next i

' * Instead of the above lines, I want to use
' * The ADC.Channels array to form the ch() array
' * That is necessary for the following function.
' * TDR
Cnt = 0
For i = 0 To ADCLASTCHANNEL
    If ADC.Channels(i) = True Then
        chans(Cnt) = i + ADCOffset
        gains(Cnt) = ADC.Gain
        Cnt = Cnt + 1
    End If
Next i
' * the cnt variable should now contain the number
' * of items in the ch and gain arrays, and can be
' * used for scansize.

' * set active channels and gains for each channel
' * in the daqbook.

er = VBdaqAdcSetScan(chans(), gains(), Cnt)

Freq = ADC.Freq
er = VBdaqAdcSetFreq(Freq)

'Set the trigger parameters
If ADC.trigger = MANUALTRIG Then
    TriggerConst = DtsSoftware
End If
er = VBdaqAdcSetTrig(TriggerConst, False, 0, 0, 0)

'Setup the background acquisition to place new readings
'into the array dataArray in the background. The other parameters
'provide infinite cycling through this array.

' scanSize = endChan - stChan + 1
ScanSize = Cnt
MaxScansInBuf = Int(ARRAYSIZE / ScanSize)
MaxBufIndex = MaxScansInBuf * ScanSize
er = VBdaqAdcRdNBack(DataArray(), MaxScansInBuf, True, False)

'The collection and storage of the data happens in the
'EmrADC_Timer event under the Timer control, in the form frmADC.
' * These two lines set the timer interval to 50 ms, and
' * enable the timer.
frmADC!tmradc.interval = 50
frmADC!tmradc.Enabled = True

```

```

    '* if it's setup for a manual trigger, then trigger
    '* the data acquisition NOW.

    If ADC.trigger = MANUALTRIG Then
        er = VBdaqAdcSoftTrig()
    End If
Case ADCCUSTOM
    '* set frequency to 10,000
    ADCSetFreq (ADFREQ10000)
    '* Output reset pulse to reset buffer address
    ADCReset
    '* start timer, wait for done signal.
    frmADC!tmrwait.Enabled = True
    '* write caption to indicate status
    frmMain!lblStatus.Caption = "Waiting for trigger"
    '* refresh the form so caption is changed
    frmMain.Refresh
    '* Send the start signal to the custom A/D
    ADCStartCustom

End Select
End Sub

Sub ADCStartCustom()
    Dim retval As Integer
    '* send reset signal
    '* reconfigure port for output
    retval = cbDConfigPort(0, FIRSTPORTCH, DIGITALOUT)
    '* Pulse the output
    retval = cbDBitOut(0, FIRSTPORTA, 21, 1)
    retval = cbDBitOut(0, FIRSTPORTA, 21, 0)
    '* Reconfigure port for input
    retval = cbDConfigPort(0, FIRSTPORTCH, DIGITALIN)

End Sub

Sub SetADC()
    Dim i As Integer
    '* close ADC before implementing new settings
    ADCClose

    ADCHardwareType = ADCList.Value(SetupData.ADChwtype)

    Select Case ADCHardwareType
    Case ADCDAQBOOK
        ADC.Gain = GainList.Value(SetupData.Gainindex)
        ADC.Freq = SetupData.SampRate
    Case ADCCUSTOM
        '* set gain to 1
        ADC.Gain = GainList.Value(0)
        SetupData.Gainindex = 0
        '* set the frequency to either 10,000, 5,000, 2500, or 1250.
        For i = 0 To 3
            If SetupData.SampRate <= 10000 / 2 ^ i Then
                ADC.Freq = 10000 / 2 ^ i
            End If
        Next i
        SetupData.SampRate = ADC.Freq

    End Select

    '* Set the filename for the A/D conversion from the text box
    ADC.filename = SetupData.filename
    ADC.numscans = SetupData.Duration * SetupData.SampRate
    ADC.trigger = TriggerList.Value(SetupData.Trigsrc)
    Select Case SetupData.Heaters.method
    Case ALLACTIVE
        For i = 0 To ADCLASTCHANNEL
            ADC.Channels(i) = True
        Next i
    End Select

```

```

        ADCGroups(i \ 16) = True
    Next i
Case RANGEACTIVE
    For i = 0 To ADCLASTCHANNEL
        ADC.Channels(i) = False
        ADCGroups(i \ 16) = False
    Next i

    For i = SetupData.Heaters.Range(0) To SetupData.Heaters.Range(1)
        ADC.Channels(i) = True
        ADCGroups(i \ 16) = True
    Next i

Case SPECIFIC
    '* set groups initially to false
    For i = 0 To 5
        ADCGroups(i) = False
    Next i

    For i = 0 To ADCLASTCHANNEL
        ADC.Channels(i) = SetupData.Heaters.List(i)
        If SetupData.Heaters.List(i) = True Then
            ADCGroups(i \ 16) = True
        End If
    Next i
End Select

ADCOffset = SetupData.ADCOffset

'* initialize ADC hardware with new settings
ADCInit

End Sub

Sub ADCClose()
    '* close the daqbook. But first check if the program is running
    '* with or without hardware.

    If ADCNOHARDWARE = False Then
        VBdaqAdcStopBack
        VBdaqClose
    End If
    Unload frmADC
End Sub

Sub ADCDefine()
    Dim i
    '*****
    '* Initialize Variables
    '*****

    '* Define trigger options for drop-down box
    TriggerList.Num = 1
    TriggerList.Value(0) = MANUALTRIG
    TriggerList.text(0) = "Manual"

    '* Define ADC options
    ADCList.Num = 3
    ADCList.Value(0) = ADCNOHARDWARE
    ADCList.text(0) = "No Hardware"
    ADCList.Value(1) = ADCDAQBOOK
    ADCList.text(1) = "DaqBook"
    ADCList.Value(2) = ADCCUSTOM
    ADCList.text(2) = "Custom A/D"

    '* Make sure all ADC channels are turned off.
    For i = 0 To ADCLASTCHANNEL

```

```

        ADC.Channels(i) = False
    Next i

    '* Define gain-options for drop-down boxes
    GainList.Num = 4
    For i = 0 To GainList.Num - 1
        GainList.Value(i) = i
        GainList.Text(i) = Str$(2 ^ GainList.Value(i))
    Next i
    ADCFirstGroup = 0
    ADCLastGroup = ADCLASTCHANNEL / 16
End Sub

Sub ADCGetData()
    '*****
    '* This subroutine handles low-level control of
    '* the A/D input. It takes a variable of type
    '* ADCType as an argument and returns the
    '* results in an array within that structure.
    '* It is only for collecting small amounts
    '* of data.
    '*****
    '* Values are returned in mV, in the buffer ADC.Buffer(0,i)
    '* where i is from 0 to ADC.numscans -1

    Dim i As Integer
    Dim er As Integer
    Dim Buffer(4000) As Integer
    Dim Cnt As Integer
    Dim TriggerConst As Integer
    Dim chans(160) As Integer
    Dim gains(160) As Integer
    Dim Freq As Single
    Dim LngDataPoint As Long

    Select Case ADCHardwareType
    Case ADCDAQBOOK
        '* set to no channel tags
        er = VBdaqAdcSetTag(0)
        '* assign the chans array
        Cnt = 0
        For i = 0 To ADCLASTCHANNEL
            If ADC.Channels(i) = True Then
                chans(Cnt) = i + ADCOffset
                gains(Cnt) = ADC.Gain
                Cnt = Cnt + 1
            End If
        Next i

        er = VBdaqAdcSetScan(chans(), gains(), Cnt)
        '* set the sampling frequency from the ADC structure
        Freq = ADC.Freq
        '* set sampling frequency in hardware
        er = VBdaqAdcSetFreq(Freq)

        '*Set the trigger parameters
        If ADC.trigger = MANUALTRIG Then
            TriggerConst = DtsSoftware
        End If
        '* set trigger in hardware
        er = VBdaqAdcSetTrig(TriggerConst, False, 0, 0, 0)
        '* save the software trigger command to the hardware
        If ADC.trigger = MANUALTRIG Then
            er = VBdaqAdcSoftTrig
        End If
        '* Read data into buffer using foreground command
        er = VBdaqAdcRdNFore(Buffer, ADC.numscans)

        '* convert hex data to voltage values
        For i = 0 To ADC.numscans - 1
            LngDataPoint = Buffer(i)

```

```

        If LngDataPoint < 0 Then
            LngDataPoint = Buffer(i) + 2 ^ 16
        End If
        ADC.Buffer(0, i) = (CDB1(LngDataPoint) / 2 ^ 16 _
            * 10000) _
            / Val(GainList.text(SetupData.Gainindex))
    Next i
End Select
End Sub

Function ADCGetWord()
    Dim fpb As Integer, fpcl As Integer, word As Integer, retval As Integer
    retval = cbDIn(ADCBOARDNUM, FIRSTPORTB, fpb)
    retval = cbDIn(ADCBOARDNUM, FIRSTPORTCL, fpcl)

    '* convert bit and nibble to unsigned 16-bit word
    ADCGetWord = (fpb + 256 * fpcl - 2048) * 2 ^ 4
End Function

Sub ADCIncrChan(steps As Integer)

    Dim retval As Integer, i As Integer, j As Integer
    '* This routine just sets the channel increment bit

    '* delay loops are added to try to get A/D cards to
    '* keep from skipping. Counting to 100 seems to work
    '* ok for the 486 laptop.
    For i = 1 To steps
        retval = cbDBitOut(ADCBOARDNUM, FIRSTPORTA, 7, 1)
        For j = 1 To 100
            Next j
        retval = cbDBitOut(ADCBOARDNUM, FIRSTPORTA, 7, 0)
        For j = 1 To 100
            Next j
        Next i
    End Sub

Sub ADCInit()
    Dim i As Integer, j As Integer
    Dim retval As Integer

    '* initialize daqbook hardware, if hardware is present
    Select Case ADCHardwareType
        Case ADCDAQBOOK
            retval = VBdaqInit(LPT1, 7)
            '* Set up for unipolar input
            retval = VBdaq200SetMode(0, 0, 0)
            '* set parallel port protocol
            retval = VBdaqSetProtocol(ADCPPI)

        Case ADCCUSTOM
            '* configure digital i/o port
            '* FIRSTPORTA, 8 bits, is the digital output channels
            retval = cbDConfigPort(0, FIRSTPORTA, DIGITALOUT)
            '* FIRSTPORTB, 8 bits, is the first 8 bits of the 12 Bit word.
            retval = cbDConfigPort(0, FIRSTPORTB, DIGITALIN)
            '* FIRSTPORTCL is the last 4 bits of the 12 bit word
            retval = cbDConfigPort(0, FIRSTPORTCL, DIGITALIN)
            '* FIRSTPORTCH is 4 bits, the first of which is the DONE bit, second
            '* is the start bit. Initially configure for input
            retval = cbDConfigPort(0, FIRSTPORTCH, DIGITALIN)

    End Select
End Sub

Sub updatePercent(numscans, i)
    Static oldPC As Long, PercentComplete As Long
    '* Display the percent complete
    oldPC = PercentComplete

```

```

PercentComplete = CLng(i) * 100 / (numscans - 1)
If Not oldPC = PercentComplete Then
    frmMain!lblStatus.Caption = "Downloading " + Str(PercentComplete) + "%"
End If
End Sub

Sub ADCReadCustom()
    Dim i As Integer, j As Integer, k As Integer, channel As Integer, skip As Integer
    Dim ScanLine As String
    frmADC!tmrwait.Enabled = False
    frmMain!lblStatus.Caption = "Downloading ..."
    frmMain.Refresh
    ADCReset

    '* define an increment size for incrementing the
    '* reading of the 10kHz data
    For k = 0 To 3
        If ADC.Freq = 10000 / 2 ^ k Then
            skip = 2 ^ k
        End If
    Next k

    For k = ADCFirstGroup To ADCLastGroup
        '* perform following steps only if these groups are active
        If ADCGroups(k) = True Then
            ADCSetGroup (k)
            ADCReset
            For j = 0 To ADC.numscans - 1
                For i = 0 To 15
                    channel = i + k * 16
                    ADCBuffer(j, channel) = ADCGetWord
                    ADCIncrChan (1)
                Next i
                '* skip ahead (skip - 1) points in time
                ADCIncrChan ((skip - 1) * 16)
                Call updatePercent((ADCLastGroup - ADCFirstGroup) * ADC.numscans, ADC.numscans *
(k - ADCFirstGroup))
                DoEvents
            Next j
        End If
    Next k

    frmMain!lblStatus.Caption = "Writing to Disk ..."

    For j = 0 To ADC.numscans - 1
        For i = 0 To ADCLASTCHANNEL
            If ADC.Channels(i) = True Then
                '* Write datapoint
                Put ADC.FileNumber, , ADCBuffer(j, i)
            End If
        Next i
        DoEvents
    Next j

    ADCStop
End Sub

Sub ADCReset()
    Dim retval As Integer
    '* send reset signal
    retval = cbDBitOut(0, FIRSTPORTA, 6, 1)
    retval = cbDBitOut(0, FIRSTPORTA, 6, 0)
End Sub

Function ADCSetGroup(Group As Integer)
    Dim bit(4) As Integer, i As Integer, retval As Integer
    For i = 3 To 0 Step -1

```

```

    If Group >= 2 ^ i Then
        Group = Group - 2 ^ i
        bit(i) = 1
    Else
        bit(i) = 0
    End If
    retval = cbDBitOut(0, FIRSTPORTA, i, bit(i))

Next i
ADCGroupNum = Group

End Function

Sub ADCStop()
    '* Halt background data collection.
    Select Case ADCHardwareType
        Case ADCDAQBOOK
            er = VBdaqAdcStopBack()
            '* Shut off data collection timer
            frmADC!tmradc.Enabled = False

        Case ADCCUSTOM
            '* shut off the wait timer
            frmADC!tmrwait.Enabled = False
            ADCReset
    End Select
    Close ADC.FileNumber

    '* Notify the main form that data acquisition is
    '* finished.
    Call frmMain.ADCDone

End Sub

Public Sub ADCTimerSub()
    '* This is the code that the timer tmrADC calls every
    '* time interval.

    '***
    '* Most of this code is from the daqbook example
    '* dbk2dsk.
    '*****

    Dim active As Integer
    Dim BufferOverflow As Integer
    Dim Unprocessed As Long
    Dim Ints As Long
    Dim FirstBufSize As Long
    Dim SecondBufSize As Long
    Dim Cnt As Long
    BufferOverflow = VBdaqAdcGetBackStat(active, Cnt)
    Debug.Print cnt, bufferOverflow

    If BufferOverflow Then
        Call ADCBufferOverflow
        Exit Sub
    End If
    If Cnt > ScansProcessed Then 'New data available
        Unprocessed = Cnt - ScansProcessed
        Ints = Unprocessed * ScanSize
        If Ints > 16000 Then
            Call ADCBufferOverflow
            Exit Sub
        End If
        If Ints + ReadIndex > MaxBufIndex Then 'With array wrap-around
            FirstBufSize = MaxBufIndex - ReadIndex + 1
            er = fWrite(FHandle, dataArray(ReadIndex), FirstBufSize * 2)
            ReadIndex = 1
            SecondBufSize = Ints - FirstBufSize
            er = fWrite(FHandle, dataArray(ReadIndex), SecondBufSize * 2)
            ReadIndex = ReadIndex + SecondBufSize
        End If
    End If

```



```

Else
    'W/O array wrap-around
    er = fWrite(FHandle, dataArray(ReadIndex), Ints * 2)
    ReadIndex = ReadIndex + Ints
End If
End If
ScansProcessed = Cnt

'PercentCompleteText.Text = 100 * ScansProcessed / ADC.NumScans

'If the desired number of scans is exceeded, stop the acquisition.
If ScansProcessed >= ADC.numscans Then
    Call ADCStop
End If

End Sub

Sub ADCBufferOverflow()
    er = VBdaqAdcStopBack()
    MsgBox "Memory Buffer Overflow..Computer too slow writing to disk.." + Chr$(13) +
Chr$(10) + "Scan rate is too Fast.."
    Close ADC.FileNameNumber
    frmADC!tmradc.Enabled = False

End Sub

Function ADCSetFreq(Freq As Integer)
    Dim bit(4 To 5) As Integer, i As Integer, retval As Integer
    Select Case Freq
        Case ADFREQ10000
            bit(4) = 0
            bit(5) = 0
        Case ADFREQ5000
            bit(4) = 1
            bit(5) = 0
        Case ADFREQ2500
            bit(4) = 0
            bit(5) = 1
        Case ADFREQ1250
            bit(4) = 1
            bit(5) = 1
    End Select
    For i = 4 To 5
        retval = cbDBitOut(0, FIRSTPORTA, i, bit(i))
    Next i
End Function

```

### **G.1.2. Listing of MODSHARE.BAS**

```

Attribute VB_Name = "modShared"
Option Explicit
'*****
'* This module contains general declarations
'* and procedures that are common to several of
'* these programs.
'*****
'*****
'* TYPE DECLARATIONS
'*****

Public Type ListType
    Value(16) As Integer
    text(16) As String
    Num As Integer
End Type

```

```

Public Type HeaterType
    '* method of selecting heaters
    method As Integer
    '* array containing specific heaters to sample
    List(160) As Boolean
    '* array containing lowest and highest heater
    '* to sample
    Range(2) As Integer
    '* number of heaters to be sampled
    Num As Integer
End Type

'* Data Type to hold calibration setup information
Type CalSetupType
    '* Heater Output threshold voltage
    Vthresh As Double
    '* How fast Vcmd will increase/decrease
    Slew As Double
    '* following was the previous definition
    '* How large a step Vcmd will take
    '* however, this variable is now used for the
    '* number of Vcmd steps per second
    Vstep As Double
    '* Vcmd to start at
    Vmin As Double
    '* Maximum Vcmd to attain
    '* Now used as the maximum difference between
    '* high and low Vcmd when doing the bisection
    '* method
    Vmax As Double
    '* How many times to repeat calibration cycle
    '* for each point
    Rep As Double
    '* number of scans to average for each vcmd step
    numscans As Double
    '* The rate to gather these scans
    ScanRate As Double
    '* defined in order to preserve compatibility
    '* with other setupdata type
    SampRate As Double
    '* Temperature that bath will be held at
    temp As Double
    '* Index of the gain value in the GainList
    Gainindex As Integer
    '* Information on which heaters to sample
    Heaters As HeaterType
    '* Comments
    Comments As String
    '* filename for saving and loading
    FileName As String
    '* hardwaretype variable to preserve compatibility
    ADChwtype As Integer
    ADCOffset As Integer
    Duration As Double
    Trigsrc As Integer
End Type

Public Type CalType
    '* calibration temperature
    temp As Double
    '* array of Vcmd values for each heater element
    Vcmd(160) As Double
    Comments As String
    Date As Date
End Type

Public Position(-1 To 8, -1 To 8) As Integer
'*****
'* CONSTANTS
'*****

```

```

'* Data File Format Constants
Public Const TEXTFORMAT = 0
Public Const BINFORMAT = 1

'* heater method constants
Public Const ALLACTIVE = 0
Public Const RANGEACTIVE = 1
Public Const SPECIFIC = 2
'* Heater area in cm^2
Public Const AvgHeaterArea = 0.000625
Public Const AvgHeaterArea = 0.0007466
Public Function ChgExt(FileName1 As String, Extension As String) As String
    '* return a string with a different file extension
    Dim Lngth As Integer
    Dim Seg As String
    Dim i As Integer

    ChgExt = FileName1
    '* find the total length of the string
    Lngth = Len(ChgExt)
    '* see if any of the last four characters are
    '* periods
    If Lngth > 3 Then
        For i = Lngth To Lngth - 3 Step -1
            Seg = Mid$(ChgExt, i, 1)
            If Seg = "." Then
                ChgExt = Left$(ChgExt, i - 1)
            End If
        Next i
        '* add on new extension
        ChgExt = ChgExt + Extension
    End If

End Function

Sub WriteCalSetup(SetupOut As CalSetupType, FileNumber As Integer)
    Dim i As Integer
    '* write all members of the SetupData structure
    '* to file with number FileNumber, in binary format.
    Write #FileNumber, SetupOut.Vthresh
    Write #FileNumber, SetupOut.Slew
    '* Step rate for stepping through new bisection method steps
    Write #FileNumber, SetupOut.Vstep
    Write #FileNumber, SetupOut.Vmin
    '* Vmax is now the Delta-V value in the bisection method
    Write #FileNumber, SetupOut.Vmax
    '* set this unused variable to the ADCOffset,
    '* so that it can be saved in the setup file
    SetupOut.Rep = SetupOut.ADCOffset
    Write #FileNumber, SetupOut.Rep
    '* Scans per data point
    Write #FileNumber, SetupOut.numscans
    '* Hardware Scan Rate
    Write #FileNumber, SetupOut.ScanRate
    Write #FileNumber, SetupOut.Gainindex
    Write #FileNumber, SetupOut.Heaters.method
    Write #FileNumber, SetupOut.Heaters.Range(0)
    Write #FileNumber, SetupOut.Heaters.Range(1)
    Write #FileNumber, SetupOut.Heaters.Num
    For i = 0 To ADCLASTCHANNEL
        If SetupOut.Heaters.List(i) = True Then
            Write #FileNumber, i
        End If
    Next i
    Write #FileNumber, SetupOut.Comments
End Sub

Sub ReadCalSetup(CalSetupIn As CalSetupType, FileNumber As Integer)
    '* read all the values of SetupData into a variable
    Dim HeaterIndex As Integer
    Dim i As Integer

```

```

    Input #FileNumber, CalSetupIn.Vthresh
    Input #FileNumber, CalSetupIn.Slew
    Input #FileNumber, CalSetupIn.Vstep
    Input #FileNumber, CalSetupIn.Vmin
    Input #FileNumber, CalSetupIn.Vmax
    Input #FileNumber, CalSetupIn.Rep
    '* set the adcoffset to the unused variable in the setup file.
    CalSetupIn.ADCOffset = CalSetupIn.Rep
    Input #FileNumber, CalSetupIn.numscans
    Input #FileNumber, CalSetupIn.ScanRate
    Input #FileNumber, CalSetupIn.Gainindex
    Input #FileNumber, CalSetupIn.Heaters.method
    Input #FileNumber, CalSetupIn.Heaters.Range(0)
    Input #FileNumber, CalSetupIn.Heaters.Range(1)
    Input #FileNumber, CalSetupIn.Heaters.Num
    For i = 0 To CalSetupIn.Heaters.Num - 1
        Input #FileNumber, HeaterIndex
        CalSetupIn.Heaters.List(HeaterIndex) = True
    Next i
    Input #FileNumber, CalSetupIn.Comments
End Sub

Sub ReadCal(ByRef CalIn As CalType, FileNumber As Integer)
    Dim i As Integer
    Input #FileNumber, CalIn.temp
    Input #FileNumber, CalIn.Date
    Input #FileNumber, CalIn.Comments
    For i = 0 To ADCLASTCHANNEL
        Input #FileNumber, CalIn.Vcmd(i)
    Next i
End Sub

Sub WriteCal(CalOut As CalType, FileNumber As Integer)
    Dim i As Integer
    Write #FileNumber, CalOut.temp
    Write #FileNumber, CalOut.Date
    Write #FileNumber, CalOut.Comments
    For i = 0 To ADCLASTCHANNEL
        Write #FileNumber, CalOut.Vcmd(i)
    Next i
End Sub

```

### **G.1.3. Listing of MODCONVE.BAS**

```

Attribute VB_Name = "modConvert"
Option Explicit
Type binary
    Value(96) As Byte
End Type
Type fluxtype
    Value(96) As Double
End Type

Dim offset As Double
Public formatnum As Integer
Const STANDARD = 0
Const MATRICES = 1
Const GNUPLOT = 2
Const MPEG = 3
Const IMAGETOOL = 4
Const MATRIXFILES = 5
Const AVGFLUX = 6
Const TIMEAVG = 7
Const RMS = 8
Const PROBDIST = 9
Const FFT = 10
Const BOIL = 11
Const SAMPLE = 12

```

```

Sub BitRemap(array() As Byte, output() As Byte, _
heaterlist() As Integer, setupin As SetupType)
    Dim k As Integer, j As Integer, byteVal As Integer, bitVal(100) As Byte
    Dim NewByteVal(100) As Byte, x As Integer, y As Integer
    '* Decode entire heater array

    For j = 0 To setupin.Heaters.Num - 1
        NewByteVal(heaterlist(j)) = array(j)
    Next j
    NewByteVal(96) = 0

    '* form a binary array variable which will be written to the
    '* output file.
    k = 0
    For y = 8 To -1 Step -1
        For x = -1 To 8
            '* subtract 1 from "position" result because it starts at
            '* heater 1, not heater 0
            output(k) = NewByteVal(Position(x, y) - 1) * 255
            k = k + 1
        Next x
    Next y

End Sub

Sub CalcBoilFunc(ByRef BoilFunc() As Byte, setupin As SetupType, ByRef Flux() As fluxtype, _
k() As Integer, TimeStep As Double, FluxThreshold As Double, _
dqdtthreshold As Double, d2qdt2threshold As Double, ByRef dqdt() As Double, _
ByRef d2qdt2() As Double)

    Dim TimeStepSqrd As Double
    Dim j As Integer
    TimeStepSqrd = TimeStep ^ 2
    For j = 0 To setupin.Heaters.Num - 1
        '* Only values from index 1 and up are valid.
        '* Absolute values are used for the following, so threshold can be taken
        '* estimate first derivative with centered difference
        dqdt(j) = Abs(Flux(k(2)).Value(j) - Flux(k(0)).Value(j)) / (2 * TimeStep)
        '* estimate second derivative with centered difference
        d2qdt2(j) = Abs(Flux(k(2)).Value(j) - 2 * Flux(k(1)).Value(j) + Flux(k(0)).Value(j)) /
TimeStepSqrd
        If (dqdt(j) > dqdtthreshold Or d2qdt2(j) > d2qdt2threshold _
Or Flux(k(1)).Value(j) > FluxThreshold) Then
            '* set bit in boiling function variable to 1
            '* NOTE: 1 is not TRUE. TRUE = -1. Comparing this value to
            '* TRUE will always result in a FALSE.
            BoilFunc(j) = 1
        Else
            BoilFunc(j) = 0
        End If
    Next j
End Sub

Sub setbit(ByRef array() As Byte, j As Integer)
    '* This function is only designed to set a bit to 1 that was
    '* previously zero.
    Dim ByteNum As Integer, BitNum As Integer
    ByteNum = j \ 8
    BitNum = j Mod 8
    array(ByteNum) = array(ByteNum) + 2 ^ BitNum

End Sub

Sub subBoilFunc()

    Dim Flux(4) As fluxtype
    Dim BoilFunc(2500) As binary

```

```

Dim k(4) As Integer, Reload As Integer, heaterlist(160) As Integer
Dim inputfilenumber As Integer, OutputFileNumber As Integer, _
    j As Integer, HeaterIndex As Integer, i As Integer, _
    numscans As Integer, output(100) As Byte, BoilFile As Integer, _
    HeaterGroup(5) As Integer, CLLength(2500) As Integer, _
    TimeHistoryFile As Integer, Index As Integer, g As Integer

Dim MinHeater As Integer, MaxHeater As Integer
Dim ResFile1 As String, ResFile2 As String, AreaFile As String, _
    sep As String, Header As String

Dim setupin As SetupType

Dim offset(160) As Double, FullScale As Double, TimeStep As Double, _
    TimeStepSqrd As Double, OutData(160) As Double, BoilSum(160) As Double, _
    BoilFlux(5) As Double, BoilFrac(5) As Double, _
    time As Double, FluxThreshold As Double, dqdtthreshold As Double, _
    d2qdt2threshold As Double, d2qdt2(100) As Double, dqdt(100) As Double, _
    LocalBoilFrac(160) As Double, RingFrac(5) As Double, RingFlux(5) As Double
Dim SumFlux(2500) As Double, SpaceFlux(100) As Double, SpaceAvg(5) As Double, _
    RingAvg(5) As Double, SumCenter As Double, LowPercent As Double, _
    HighPercent As Double, LocalNorm(-90 To 160, 160) As Double, _
    RingHigh(5) As Double, RingLow(5) As Double, RingNorm(-20 To 160, 5) As Double, _
    RingSum As Double, TotalSum As Double, TotalLow(5) As Double, _
    TotalHigh(5) As Double, TotalNorm(-20 To 160, 5) As Double
Dim Sum As Double, low(100) As Double, high(100) As Double, DistWidth(100) As Double, _
    LocalBoilFlux(100) As Double

Dim BoilCount(5) As Long, LocalBoilCount(160) As Long, _
    RingCount(5) As Long, CLTotal As Long, LocalProb(-90 To 160, 160) As Long, _
    RingProb(-20 To 160, 4) As Long, TotalProb(-20 To 160, 5) As Long

HeaterGroup(0) = 0
HeaterGroup(1) = 4
HeaterGroup(2) = 16
HeaterGroup(3) = 36
HeaterGroup(4) = 64
HeaterGroup(5) = 96

time = 0
'* insure that resistance files will be loaded the first time through
Reload = True
ResFile1 = frmConvert!txtresfile1.text
ResFile2 = frmConvert!txtresfile2.text
AreaFile = frmConvert!txtAreaFile.text
FluxThreshold = Val(frmConvert!txtfluxthreshold.text)
dqdtthreshold = Val(frmConvert!txtdqdtThreshold.text)
d2qdt2threshold = Val(frmConvert!txtd2qdt2Threshold.text)
LowPercent = Val(frmConvert!txtLowPercent.text) / 100
HighPercent = Val(frmConvert!txtHighPercent.text) / 100

'* define column seperator - chr$(9) is a Tab
sep = Chr$(9)
'* open input file and load setup information
Call openinputfile(setupin, inputfilenumber, frmConvert!txtinputfile.text)
Call OpenOutputFile(formatnum, OutputFileNumber, frmConvert!txtoutputfile.text)
Call subReadOffset(offset, frmConvert!txtoffsetfile.text)
Call subHeaterList(setupin)
BoilFile = FreeFile
Open frmConvert!txtBoilFile.text For Output As BoilFile

TimeHistoryFile = FreeFile
Open frmConvert!txtTimeHistoryFile.text For Output As TimeHistoryFile
HeaterIndex = 0
For i = 0 To ADCLASTCHANNEL
    If setupin.Heaters.List(i) = True Then
        heaterlist(HeaterIndex) = i
        HeaterIndex = HeaterIndex + 1
    End If

```

```

Next i

FullScale = SetFullScale(setupin.ADChwtype)

TimeStep = 1 / setupin.SampRate
TimeStepSqrd = TimeStep ^ 2
numscans = setupin.Duration * setupin.SampRate

'* set the initial value of the index variables
k(0) = 0
k(1) = 1
k(2) = 2

For i = 0 To numscans - 1
    time = TimeStep * i
    '* load voltage on heaters for one timestep into
    '* OutData matrix
    Call subLoadStep(setupin, inputfilenumber, FullScale, OutData)
    '* calculate power from voltage using resistance files
    '* place result in Flux matrix
    Call CalcFlux(OutData, Flux(k(2)).Value, offset, ResFile1, ResFile2, AreaFile, _
        setupin.Temp, setupin.Heaters.Num, Reload)
    Call CalcBoilFunc(BoilFunc(i).Value, setupin, Flux, k, TimeStep, _
        FluxThreshold, dqdtthreshold, d2qdt2threshold, dqdt, d2qdt2)
    For j = 0 To setupin.Heaters.Num
        If BoilFunc(i).Value(j) = 1 Then
            '* sum the number of boiling occurrences
            LocalBoilCount(j) = LocalBoilCount(j) + 1
            '* sum the boiling heat fluxes
            BoilSum(j) = BoilSum(j) + Flux(k(1)).Value(j)
            '* sum the probability distributions
            '* assign an index to a probability bin
            Index = Int(Flux(k(1)).Value(j))
            '* add one to the probability bin
            LocalProb(Index, j) = LocalProb(Index, j) + 1

        End If
        '* sum the space-averaged, time-resolved component
        SumFlux(i) = SumFlux(i) + Flux(k(1)).Value(j)
        '* sum the time-averaged, space-resolved component
        SpaceFlux(j) = SpaceFlux(j) + Flux(k(1)).Value(j)
    Next j
    '* rotate the index variable
    k(3) = k(0)
    k(0) = k(1)
    k(1) = k(2)
    k(2) = k(3)

    Call subContactLine(BoilFunc(i).Value, CLLength(i), heaterlist, setupin)
    CLTotal = CLTotal + CLLength(i)
    Call subUpdateStatus(numscans, i)
    DoEvents
Next i

'* calculate probability density parameters
'* calculate the width of the distribution
For j = 0 To setupin.Heaters.Num - 1
    Sum = 0
    For i = -20 To 160
        '* normalize local distribution interval
        If LocalBoilCount(j) > 0 Then
            LocalNorm(i, j) = CDbl(LocalProb(i, j)) / CDbl(LocalBoilCount(j))
        Else
            LocalNorm(i, j) = 0
        End If
        Sum = Sum + LocalNorm(i, j)
        If Sum >= LowPercent Then
            low(j) = i + 1
            Exit For
        End If
    Next i

```

```

Next i
Sum = 0
For i = 160 To -20 Step -1
    Sum = Sum + LocalNorm(i, j)
    If Sum >= HighPercent Then
        high(j) = i - 1
        Exit For
    End If
Next i
DistWidth(j) = high(j) - low(j)
If high(j) - low(j) < 0 Then DistWidth(j) = 0
If LocalBoilCount(j) > 0 Then
    LocalBoilFlux(j) = BoilSum(j) / LocalBoilCount(j)
Else
    LocalBoilFlux(j) = 0
End If
Next j

For i = 0 To numscans - 1
    Call BitRemap(BoilFunc(i).Value, output, heaterlist, setupin)
    '* add three because the corner heaters are also included
    For j = 0 To 99
        Put #OutputFileNumber, , output(j)
        output(j) = 0
    Next j
    '* calculate the average of the boiling function over the center
    '* four heaters
    For j = 0 To 3
        SumCenter = SumCenter + BoilFunc(i).Value(heaterlist(j))
    Next j
    '* divide the sum by 4 to get the average
    SumCenter = SumCenter / 4
    '* Print the time and the average value to a data file
    Print #TimeHistoryFile, Str(i * TimeStep) + sep + Format(SumCenter, "scientific")
    '* reset the summation variable to zero
    SumCenter = 0
    '* all other system tasks to execute
    DoEvents
Next i

'* obtain summations
For i = 0 To 4
    For j = HeaterGroup(i) To HeaterGroup(i + 1) - 1
        RingCount(i) = RingCount(i) + LocalBoilCount(j)
        LocalBoilFrac(j) = LocalBoilCount(j) / numscans
        RingAvg(i) = RingAvg(i) + SpaceFlux(j)
        '* sum up probability distributions for the rings
        For g = -20 To 160
            RingProb(g, i) = RingProb(g, i) + LocalProb(g, j)
        Next g
        If LocalBoilCount(j) > 0 Then
            RingFlux(i) = RingFlux(i) + BoilSum(j) / LocalBoilCount(j)
        End If
    Next j

    '* sum the total count of boiling events
    BoilCount(i) = BoilCount(i) + RingCount(i)

    '* normalize the ring distributions for number of boiling occurrences
    '* in that ring, and calculate high and low
    RingSum = 0
    TotalSum = 0
    For g = -20 To 160
        '* normalize ring distribution
        If RingCount(i) > 0 Then
            RingNorm(g, i) = RingProb(g, i) / RingCount(i)
        End If
        TotalProb(g, i) = TotalProb(g, i) + RingProb(g, i)
        If BoilCount(i) > 0 Then
            TotalNorm(g, i) = TotalProb(g, i) / BoilCount(i)
        End If
    Next g

```



```

End If
TotalProb(g, i + 1) = TotalProb(g, i)
' * sum the distribution at each interval
RingSum = RingSum + RingNorm(g, i)
TotalSum = TotalSum + TotalNorm(g, i)
' * check to see if the probability limit has been reached
If RingSum <= LowPercent Then
    ' * set the lower limit value
    RingLow(i) = g + 1
End If
If RingSum <= 1 - HighPercent Then
    ' * set upper limit value
    RingHigh(i) = g + 1
End If
If TotalSum <= LowPercent Then
    TotalLow(i) = g + 1
End If
If TotalSum <= 1 - HighPercent Then
    TotalHigh(i) = g + 1
End If

Next g
' * The partition indexes (k) mark the BOTTOM of the interval containing
' * the heat fluxes. Therefore, the NEXT interval (k+1) is the
' * partition between the lower and higher probabilities.

' * calculate the sumation for average boiling flux over
' * section of heaters
BoilFlux(i) = BoilFlux(i) + RingFlux(i)
' * add current count and flux summations to the summation
' * for the next step
SpaceAvg(i) = SpaceAvg(i) + RingAvg(i)
BoilCount(i + 1) = BoilCount(i)
BoilFlux(i + 1) = BoilFlux(i)
SpaceAvg(i + 1) = SpaceAvg(i)

' * divide the sumations to obtain the averages
BoilFlux(i) = BoilFlux(i) / HeaterGroup(i + 1)
RingFlux(i) = RingFlux(i) / (HeaterGroup(i + 1) - HeaterGroup(i))
BoilFrac(i) = BoilCount(i) / numscans / HeaterGroup(i + 1)
RingFrac(i) = RingCount(i) / numscans _
    / (HeaterGroup(i + 1) - HeaterGroup(i))
RingAvg(i) = RingAvg(i) / (HeaterGroup(i + 1) - HeaterGroup(i)) / numscans
SpaceAvg(i) = SpaceAvg(i) / HeaterGroup(i + 1) / numscans
Next i

' * Calculate boiling heat flux
Print #BoilFile, "Local Boiling Fraction"
Call subWriteHeater(setupin, LocalBoilFrac, BoilFile, heaterlist)
Print #BoilFile, "Local Boiling Heat Flux"
Call subWriteHeater(setupin, LocalBoilFlux, BoilFile, heaterlist)
Print #BoilFile, "Boiling Heat Flux"
Print #BoilFile, "Total" + sep + "Ring"
For j = 0 To 4
    Print #BoilFile, Format(BoilFlux(j), "scientific") + sep _
        + Format(RingFlux(j), "scientific")
Next j
Print #BoilFile, "Boiling Fraction"
For j = 0 To 4
    Print #BoilFile, Format(BoilFrac(j), "scientific") + sep _
        + Format(RingFrac(j), "scientific")
Next j
Print #BoilFile, "Average Heat Flux"
For j = 0 To 4
    Print #BoilFile, Format(SpaceAvg(j), "scientific") + sep _
        + Format(RingAvg(j), "scientific")
Next j
Print #BoilFile, "Low Heat Flux"
For j = 0 To 4

```

```

        Print #BoilFile, Format(TotalLow(j), "scientific") + sep _
            + Format(RingLow(j), "scientific")
    Next j
    Print #BoilFile, "High Heat Flux"
    For j = 0 To 4
        Print #BoilFile, Format(TotalHigh(j), "scientific") + sep _
            + Format(RingHigh(j), "scientific")
    Next j

    Print #BoilFile, "Contact Line Length" + sep + Format(CLTotal / numscans, "scientific")
    Print #BoilFile, "Heat Flux Threshold" + sep + Format(FluxThreshold, "scientific")
    Print #BoilFile, "dq/dt Theshold" + sep + Format(dqdtthreshold, "scientific")
    Print #BoilFile, "d2q/dt2 Threshold" + sep + Format(d2qdt2threshold, "scientific")

    Close inputfilenumber
    Close OutputFileNumber
    Close BoilFile
    Close TimeHistoryFile
End Sub

Sub subContactLine(BoilFunc() As Byte, CLLength As Integer, heaterlist() As Integer, ByRef
setupin As SetupType)
    Dim j As Integer, x As Integer, y As Integer
    Dim LastPoint As Byte, NewPoint As Byte
    Dim array(160) As Byte

    For j = 0 To setupin.Heaters.Num - 1
        array(heaterlist(j)) = BoilFunc(j)
    Next j
    array(96) = 0

    '* form a binary array2 variable which will be written to the
    '* output file.
    For y = 0 To 7
        LastPoint = array(Position(0, y) - 1)
        For x = 1 To 7

            '* subtract 1 from "position" result because it starts at
            '* heater 1, not heater 0
            NewPoint = array(Position(x, y) - 1)
            If Not NewPoint = LastPoint Then
                CLLength = CLLength + 1
            End If

            LastPoint = NewPoint
        Next x
    Next y

    For x = 0 To 7
        LastPoint = array(Position(x, 0) - 1)
        For y = 1 To 7
            NewPoint = array(Position(x, y) - 1)
            If Not NewPoint = LastPoint Then
                CLLength = CLLength + 1
            End If
            LastPoint = NewPoint
        Next y
    Next x

End Sub

Sub subReadOffset(offset() As Double, filename As String)
    Dim calOffset As CalType
    Dim CalSetup As CalSetupType
    Dim freefilenum As Integer
    Dim i As Integer

    freefilenum = FreeFile
    Open filename For Input As freefilenum
    Call ReadCalSetup(CalSetup, freefilenum)
    Call ReadCal(calOffset, freefilenum)

```

```

Close freefilenum
For i = 0 To CalSetup.Heaters.Num - 1
    offset(i) = calOffset.Vcmd(i)
Next i

End Sub

Function SetFullScale(HWType As Integer)

    '* set the voltage scaling to the correct value
    If ADCList.Value(HWType) = ADCDAQBOOK Then
        SetFullScale = 10
    ElseIf ADCList.Value(HWType) = ADCCUSTOM Then
        SetFullScale = 12
    Else
        SetFullScale = 10
    '* in case we read old files that don't have the extra setup data
    End If

End Function

Sub subHeaterList(ByRef setupin As SetupType)
    '* the following code just converts all the other
    '* heater specification methods to the list method.
    '* This is just a work-around for the way this routine was
    '* originally written.

    Dim i As Integer
    Select Case setupin.Heaters.method
        Case ALLACTIVE
            For i = 0 To ADCLASTCHANNEL
                setupin.Heaters.List(i) = i
                setupin.Heaters.Num = ADCLASTCHANNEL + 1
            Next i
        Case RANGEACTIVE
            setupin.Heaters.Num = setupin.Heaters.Range(1) _
                - setupin.Heaters.Range(0) + 1
            For i = 0 To ADCLASTCHANNEL
                If i >= setupin.Heaters.Range(0) _
                    And i <= setupin.Heaters.Range(1) Then
                    setupin.Heaters.List(i) = True
                Else
                    setupin.Heaters.List(i) = False
                End If
            Next i
        End Select
    End Sub

Sub subAvgFlux()
    Dim Reload As Integer, numscans As Integer, inputfilenumber As Integer, _
        OutputFileNumber As Integer, i As Integer, j As Integer
    Dim FullScale As Double, time As Double, OutData(160) As Double, _
        Flux(160) As Double, SumPower As Double, TotalSum As Double, _
        TimeStep As Double, AveragePower As Double, offset(160) As Double
    Dim ResFile1 As String, ResFile2 As String, AreaFile As String, _
        sep As String, Formatted As String, ScanLine As String
    Dim setupin As SetupType

    time = 0
    TotalSum = 0
    SumPower = 0
    '* insure that resistance files will be loaded the first time through
    Reload = True
    ResFile1 = frmConvert!txtresfile1.text
    ResFile2 = frmConvert!txtresfile2.text
    AreaFile = frmConvert!txtAreaFile.text
    '* define column separator - chr$(9) is a Tab
    sep = Chr$(9)

```

```

'* open input file and load setup information
Call openinputfile(setupin, inputfilenumber, frmConvert!txtinputfile.text)
Call OpenOutputFile(formatnum, OutputFileNumber, frmConvert!txtoutputfile.text)

Call subHeaterList(setupin)
Call subReadOffset(offset, frmConvert!txtoffsetfile.text)
FullScale = SetFullScale(setupin.ADChwtype)

TimeStep = 1 / setupin.SampRate
numscans = setupin.Duration * setupin.SampRate

For i = 0 To numscans - 1
    time = TimeStep * i
    '* load voltage on heaters for one timestep into
    '* OutData matrix
    Call subLoadStep(setupin, inputfilenumber, FullScale, OutData)

    '* calculate power from voltage using resistance files
    Call CalcFlux(OutData, Flux, offset, ResFile1, ResFile2, AreaFile, _
        setupin.Temp, setupin.Heaters.Num, Reload)

    For j = 0 To setupin.Heaters.Num - 1
        '* sum the power from each heater
        SumPower = SumPower + Flux(j)
    Next j
    AveragePower = SumPower / setupin.Heaters.Num
    '* print the average to the outputfile
    '* format the average for output
    Formatted = Format(AveragePower, "scientific")
    '* compute the total sum for an overall average heat flux
    '* value
    TotalSum = TotalSum + AveragePower
    '* add the time stamp to the beginning of the line
    ScanLine = Str$(time) + sep + Formatted
    '* output to a file
    Print #OutputFileNumber, ScanLine
    '* reset sum value to zero
    SumPower = 0
    Call subUpdateStatus(numscans, i)
    DoEvents
Next i
'* Output, as a final value, the overall average
'* heat flux. Add two separation characters
'* to put it in the 3rd column in Excell
'* compute the average and format the average for output
Formatted = Format(TotalSum / numscans, "scientific")
frmConvert!txtavgflux.text = Formatted
'* add the time stamp to the beginning of the line
ScanLine = sep + sep + Formatted
'* output to a file
Print #OutputFileNumber, ScanLine
Close inputfilenumber
Close OutputFileNumber

End Sub

Sub subLoadStep(setupin As SetupType, inputfilenumber As Integer, FullScale As Double, ByRef
OutData() As Double)
    Dim j As Integer
    ReDim datapoint(0 To setupin.Heaters.Num - 1) As Integer
    Dim LngDataPoint As Long
    Get inputfilenumber, , datapoint
    For j = 0 To setupin.Heaters.Num - 1
        '* input one data point
        '* each datapoint is loaded as an integer
        LngDataPoint = datapoint(j)
        OutData(j) = (CDBl(LngDataPoint) / 2 ^ 16 _
            * FullScale) _
            / Val(GainList.text(setupin.Gainindex))
        If OutData(j) < 0 Then OutData(j) = OutData(j) + FullScale
    Next j

```

```

    '* OutData now contains all the heater voltages at that
    '* time step.

End Sub

Sub subProbDist()
    Dim Reload As Integer, numscans As Integer, inputfilenumber As Integer, _
        OutputFileNumber As Integer, i As Integer, j As Integer, _
        Index As Integer, Dist(-90 To 160) As Long, _
        LocalDist(-90 To 160, 160) As Long, _
        HeaterIndex As Integer, heaterlist(160) As Integer
    Dim FullScale As Double, time As Double, OutData(160) As Double, _
        Flux(160) As Double, Normalized As Double, _
        TimeStep As Double, TotalNum As Double, Sum As Double, _
        AvgData(160) As Double, rmsdata(160) As Double, Mode(160) As Double, _
        Median(160) As Double, DistWidth(160) As Double, MaxProb As Double, _
        high(160) As Double, low(160) As Double, LocalNorm(-90 To 160, 160) As Double

    Dim ResFile1 As String, ResFile2 As String, AreaFile As String, _
        sep As String, Formatted As String, ScanLine As String, lineout As String
    Dim setupin As SetupType
    Dim offset(160) As Double, LowPercent As Double, HighPercent As Double

    time = 0
    '* insure that resistance files will be loaded the first time through
    Reload = True
    ResFile1 = frmConvert!txtresfile1.text
    ResFile2 = frmConvert!txtresfile2.text
    AreaFile = frmConvert!txtAreaFile.text
    LowPercent = Val(frmConvert!txtLowPercent.text) / 100
    HighPercent = Val(frmConvert!txtHighPercent.text) / 100
    '* define column separator - chr$(9) is a Tab
    sep = Chr$(9)
    '* open input file and load setup information
    Call openinputfile(setupin, inputfilenumber, frmConvert!txtinputfile.text)
    Call OpenOutputFile(formatnum, OutputFileNumber, frmConvert!txtoutputfile.text)
    Call subReadOffset(offset, frmConvert!txtoffsetfile.text)
    Call subHeaterList(setupin)
    HeaterIndex = 0
    For i = 0 To ADCLASTCHANNEL
        If setupin.Heaters.List(i) = True Then
            heaterlist(HeaterIndex) = i
            HeaterIndex = HeaterIndex + 1
        End If
    Next i

    FullScale = SetFullScale(setupin.ADChwtype)

    TimeStep = 1 / setupin.SampRate
    numscans = setupin.Duration * setupin.SampRate

    For i = 0 To numscans - 1
        time = TimeStep * i
        '* load voltage on heaters for one timestep into
        '* OutData matrix
        Call subLoadStep(setupin, inputfilenumber, FullScale, OutData)

        '* calculate power from voltage using resistance files
        Call CalcFlux(OutData, Flux, offset, ResFile1, ResFile2, AreaFile, _
            setupin.Temp, setupin.Heaters.Num, Reload)
        Call subTimeAvg(AvgData, Flux, numscans, setupin.Heaters.Num)
        Call subRMS(rmsdata, Flux, numscans, setupin.Heaters.Num)

        For j = 0 To setupin.Heaters.Num - 1
            '* return the first integer less than or equal to (flux(j))
            '* to be used as an index in a probability distribution
            '* Therefore, the value of Index indicates the lower bound

```

```

    '* of the interval which contains the heat flux value. The
    '* lower bound will be inclusive.

    Index = Int(Flux(j))
    LocalDist(Index, j) = LocalDist(Index, j) + 1
Next j
Call subUpdateStatus(numscans, i)
DoEvents
Next i
TotalNum = CDBl(numscans) * CDBl(setupin.Heaters.Num)
'* sum the total distribution
For i = -20 To 160
    For j = 0 To setupin.Heaters.Num - 1
        Dist(i) = Dist(i) + LocalDist(i, j)
    Next j
Next i

'* write distributions to file
For i = -20 To 160
    '* normalize the distribution to the number of samples
    lineout = lineout + Str(i)
    For j = 0 To setupin.Heaters.Num - 1
        '* normalize local distribution interval
        Normalized = CDBl(LocalDist(i, j)) / CDBl(numscans)
        '* format for scientific notation output
        Formatted = Format(Normalized, "scientific")
        '* add number onto one row
        lineout = lineout + sep + Formatted
        '* assign normalized value to a local normalized array
        LocalNorm(i, j) = Normalized
    Next j
    '* normalize the total distribution and add that onto last
    '* column
    Normalized = CDBl(Dist(i)) / TotalNum
    Formatted = Format(Normalized, "scientific")
    lineout = lineout + sep + Formatted
    '* output one row
    Print #OutputFileNumber, lineout
    '* clear row
    lineout = ""
    DoEvents
Next i

'* calculate the width of the distribution
For j = 0 To setupin.Heaters.Num - 1
    Sum = 0
    For i = -20 To 160
        Sum = Sum + LocalNorm(i, j)
        If Sum >= LowPercent Then
            low(j) = i + 1
            Exit For
        End If
    Next i
    Sum = 0
    For i = 160 To -20 Step -1
        Sum = Sum + LocalNorm(i, j)
        If Sum >= HighPercent Then
            high(j) = i - 1
            Exit For
        End If
    Next i
    DistWidth(j) = high(j) - low(j)
    If high(j) - low(j) < 0 Then DistWidth(j) = 0
Next j

'* calculate mode
For j = 0 To setupin.Heaters.Num - 1
    MaxProb = 0

    For i = -20 To 160

```

```

        If LocalNorm(i, j) > MaxProb Then
            MaxProb = LocalNorm(i, j)
            Mode(j) = i
        End If
    Next i
Next j

** calculate median
For j = 0 To setupin.Heaters.Num - 1
    Sum = 0
    For i = -20 To 160
        Sum = Sum + LocalNorm(i, j)
        If Sum >= 0.5 Then
            Median(j) = i
            Exit For
        End If
    Next i
Next j
Close inputfilenumber
Close OutputFileNumber

** Check to see if file names exist, write various data files to disk
If Not frmConvert!txtmodefile = "" Then
    OutputFileNumber = FreeFile
    Open frmConvert!txtmodefile For Output As OutputFileNumber
    Call subWriteHeater(setupin, Mode, OutputFileNumber, heaterlist)
    Close OutputFileNumber
End If
If Not frmConvert!txtwidthfile.text = "" Then
    OutputFileNumber = FreeFile
    Open frmConvert!txtwidthfile.text For Output As OutputFileNumber
    Call subWriteHeater(setupin, DistWidth, OutputFileNumber, heaterlist)
    Close OutputFileNumber
End If
If Not frmConvert!txtmedianfile.text = "" Then
    OutputFileNumber = FreeFile
    Open frmConvert!txtmedianfile.text For Output As OutputFileNumber
    Call subWriteHeater(setupin, Median, OutputFileNumber, heaterlist)
    Close OutputFileNumber
End If
If Not frmConvert!txtmaxfile.text = "" Then
    OutputFileNumber = FreeFile
    Open frmConvert!txtmaxfile.text For Output As OutputFileNumber
    Call subWriteHeater(setupin, high, OutputFileNumber, heaterlist)
    Close OutputFileNumber
End If
If Not frmConvert!txtminfile.text = "" Then
    OutputFileNumber = FreeFile
    Open frmConvert!txtminfile.text For Output As OutputFileNumber
    Call subWriteHeater(setupin, low, OutputFileNumber, heaterlist)
    Close OutputFileNumber
End If

End Sub
Sub subFFT()
    Dim c(160, 2048, 3) As Double, csum(2048), f As Double, fmax As Double
    Dim FullScale As Double, OutData(160) As Double, Flux(160) As Double
    Dim offset(100) As Double
    Dim m As Integer, i As Integer, j As Integer, n As Integer, _
        heater As Integer, Reload As Integer, inputfilenumber As Integer, _
        OutputFileNumber As Integer, numscans As Integer, k As Integer, _
        LastHeater As Integer
    Dim numheaters As Integer
    Dim sep As String, ResFile1 As String, ResFile2 As String, _
        AreaFile As String, Formatted As String
    Dim setupin As SetupType
    Const Re = 0
    Const Im = 1
    Const mag = 2
    Const PI = 3.14159
    ** fft summation obtained from Mathcadd help.

```

```

Reload = True
ResFile1 = frmConvert!txtresfile1.text
ResFile2 = frmConvert!txtresfile2.text
AreaFile = frmConvert!txtAreaFile.text
numheaters = Val(frmConvert!txtNumHeaters.text)
'* define column separator - chr$(9) is a Tab
sep = Chr$(9)
'* open input file and load setup information
Call openinputfile(setupin, inputfilenumber, frmConvert!txtinputfile.text)
Call OpenOutputFile(formatnum, OutputFileNumber, frmConvert!txtoutputfile.text)

Call subHeaterList(setupin)
FullScale = SetFullScale(setupin.ADChwtype)
numscans = setupin.Duration * setupin.SampRate
'* calculate nyquist frequency
fmax = setupin.SampRate / 2

'* find value of m
For i = 2 To 16
    If 2 ^ i <= numscans And 2 ^ (i + 1) > numscans Then
        m = i
    End If
Next i
'* define n -- number of points to use in the FFT.
n = 2 ^ m

For k = 1 To n
    Call subLoadStep(setupin, inputfilenumber, FullScale, OutData)
    '* calculate power from voltage using resistance files
    Call CalcFlux(OutData, Flux, offset, ResFile1, ResFile2, AreaFile, _
        setupin.Temp, numheaters, Reload)

    For heater = 0 To numheaters - 1

        For j = 0 To n / 2
            c(heater, j, Re) = c(heater, j, Re) + Flux(heater) _
                * Cos(-2 * PI * j * k / n)
            c(heater, j, Im) = c(heater, j, Im) + Flux(heater) _
                * Sin(-2 * PI * j * k / n)
        Next j
        DoEvents
    Next heater
    Call subUpdateStatus(n, k)
Next k

'* calculate magnitudes
For j = 0 To n / 2
    For heater = 0 To numheaters - 1
        '* divide all the summations by two to obtain the final FFT values
        c(heater, j, Re) = c(heater, j, Re) / n
        c(heater, j, Im) = c(heater, j, Im) / n
        '* calculate the magnitudes of all the fft values
        c(heater, j, mag) = Sqr(c(heater, j, Re) ^ 2 + c(heater, j, Im) ^ 2)
        '* sum the magnitudes of all the heaters at each frequency
        csum(j) = csum(j) + c(heater, j, mag)
    Next heater
    '* normalize by the number of heaters
    csum(j) = csum(j) / numheaters
    '* calculate frequency for this value of j
    f = fmax * j / (n / 2)
    '* format frequency component magnitude for output
    Formatted = Format(csum(j), "scientific")
    '* write line to file
    Print #OutputFileNumber, Str(f) + sep + Formatted
    Call subUpdateStatus(n / 2, j)
Next j

Close inputfilenumber
Close OutputFileNumber

```



End Sub

```
Sub subRMS(ByRef OutData() As Double, InData() As Double, numscans As Integer, numheaters As Integer)
```

```
    Static Sum(99) As Double, SumSquare(160) As Double
```

```
    Static n As Integer
```

```
    Dim i As Integer
```

```
    Dim a As Double, b As Double, c As Double
```

```
    '* this subroutine depends on VB setting all declared variables = 0
```

```
    '* when the program starts. Other languages don't do that.
```

```
    '* sum heat fluxes for each heater
```

```
    For i = 0 To numheaters
```

```
        Sum(i) = Sum(i) + InData(i)
```

```
        SumSquare(i) = SumSquare(i) + InData(i) ^ 2
```

```
    Next i
```

```
    '* increment the scan count
```

```
    n = n + 1
```

```
    '* check to see if all the scans have been summed
```

```
    If n = numscans Then
```

```
        '* reset scan count
```

```
        n = 0
```

```
        '* set outdata equal to the average
```

```
        For i = 0 To numheaters - 1
```

```
            a = SumSquare(i) / (numscans - 1)
```

```
            b = Sum(i) ^ 2 / ((numscans - 1))
```

```
            c = a - b / numscans
```

```
            If c < 0 Then c = 0
```

```
            OutData(i) = Sqr(c)
```

```
        '* reset sum to zero
```

```
        Sum(i) = 0
```

```
        SumSquare(i) = 0
```

```
    Next i
```

```
End If
```

End Sub

```
Sub setoffset(offsetval As Double)
```

```
    offset = offsetval
```

End Sub

```
Sub subConvert()
```

```
    Dim ScanLine As String, sep As String, strDate As String, _  
        strTime As String, Comments As String, Formatted As String, _  
        BaseName As String, appendage As String, ResFile1 As String, _  
        ResFile2 As String, AreaFile As String
```

```
    Dim lngDataPoint As Long
```

```
    Dim i As Integer, j As Integer, x As Integer, y As Integer, _
```

```
        a As Integer, b As Integer, datapoint As Integer, _
```

```
        inputfilenumber As Integer, OutputFileNumber As Integer, _
```

```
        TagFileNumber As Integer, HeaterIndex As Integer, _
```

```
        heaterlist(160) As Integer, bMax As Integer, _
```

```
        numscans As Integer, LastIndex As Integer, _
```

```
        Reload As Integer, offsetfilenumber
```

```
    Dim MaxPower As Double, OutData(160) As Double, _
```

```
        time As Double, TimeStep As Double, SumPower As Double, TotalSum As Double, _
```

```
        AveragePower As Double, Flux(160) As Double, FullScale As Double, _
```

```
        Values(160) As Double
```

```
    Dim setupin As SetupType
```

```
    Dim itout(44) As Byte
```

```
    Dim offset(160) As Double
```

```
    TotalSum = 0
```

```
    '* insure that resistance files will be loaded the first time through
```

```
    Reload = True
```

```
    ResFile1 = frmConvert!txtresfile1.text
```

```
    ResFile2 = frmConvert!txtresfile2.text
```

```
    AreaFile = frmConvert!txtAreaFile.text
```

```
    Call subReadOffset(offset, frmConvert!txtoffsetfile.text)
```

```

'For i = 0 To 5
'    If optFormat(i).Value = True Then
'        FormatNum = i
'    End If
'Next i

MaxPower = 0
time = 0
ScanLine = ""
Select Case formatnum
Case STANDARD, MATRICES, MATRIXFILES, AVGFLUX, TIMEAVG, RMS
    '* define column separator - chr$(9) is a Tab
    sep = Chr$(9)
Case GNUPLOT, MPEG
    sep = CRLF
End Select

'* open input file and load setup information
Call openinputfile(setupin, inputfilenumber, frmConvert!txtinputfile.text)

Call subHeaterList(setupin)

Call OpenOutputFile(formatnum, OutputFileNumber, frmConvert!txtoutputfile.text)

FullScale = SetFullScale(setupin.ADChwtype)

'* If the textbox is non-zero, limit the number of scans to read in
numscans = setupin.Duration * setupin.SampRate
If Val(frmConvert!txtsteps.text) > 0 Then
    If numscans > Val(frmConvert!txtsteps.text) Then numscans = Val(frmConvert!txtsteps.text)
End If
TimeStep = 1 / setupin.SampRate

'* write heading with channels
'* insert first tab to skip first column
ScanLine = ScanLine + sep
HeaterIndex = 0

For i = 0 To ADCLASTCHANNEL
    If setupin.Heaters.List(i) = True Then
        ScanLine = ScanLine + Str$(i) + sep
        heaterlist(HeaterIndex) = i
        HeaterIndex = HeaterIndex + 1
    End If
Next i

Select Case formatnum
Case STANDARD
    Print #OutputFileNumber, ScanLine
End Select

ScanLine = ""

'* Step through the time steps
For i = 0 To numscans - 1
    time = TimeStep * i
    If formatnum = MATRIXFILES Then
        Call OpenMatrixFile(frmConvert!txtoutputfile.text, OutputFileNumber, i)
    End If
    '* load voltage on heaters for one timestep into
    '* OutData matrix
    Call subLoadStep(setupin, inputfilenumber, FullScale, OutData)

    '* calculate power from voltage using resistance files
    Call CalcFlux(OutData, Flux, offset, ResFile1, ResFile2, AreaFile, _
        setupin.Temp, setupin.Heaters.Num, Reload)

    Select Case formatnum
    Case MATRICES, GNUPLOT, MPEG, IMAGETOOL, MATRIXFILES
        For j = 0 To setupin.Heaters.Num - 1

```

```

    '* outdata array doesn't necessarily contain the
    '* heater information in the order of the heater numbers.
    '* therefore we assign the Values array by remapping using
    '* the heaterlist array.
    Values(heaterlist(j)) = OutData(j)
  Next j
End Select

Select Case formatnum
Case STANDARD
  For j = 0 To setupin.Heaters.Num - 1
    Formatted = Format(Flux(j), "Scientific")
    '* add channel sample to line of data
    ScanLine = ScanLine + Formatted + sep
  Next j
  '* append time index to line of data
  ScanLine = Str$(time) + sep + ScanLine
  Print #OutputFileNumber, ScanLine
  '* clear scanline
  ScanLine = ""
Case AVGFLUX
Case TIMEAVG
  '* when the last iteration is reached, outdata will contain the
  '* time-averaged heater values.
  Call subTimeAvg(Flux, Flux, numscans, setupin.Heaters.Num)
Case RMS
  Call subRMS(Flux, Flux, numscans, setupin.Heaters.Num)
Case MATRICES, GNUPLOT, MPEG, MATRIXFILES
  For x = 0 To setupin.Heaters.Num
    If Flux(x) > MaxPower Then MaxPower = OutData(x)
    If formatnum = MPEG Then
      '* inverse the data so GNUplot ranges will be right
      Flux(x) = -Flux(x)
    End If
  Next x
  Call subWriteHeater(setupin, Flux(), OutputFileNumber, heaterlist)

  '* append a line feed
  Print #OutputFileNumber, ""
Case IMAGETOOL
  Call ITWrite(OutputFileNumber)
End Select

Call subUpdateStatus(numscans, i)

If formatnum = MATRIXFILES Then
  Close OutputFileNumber
End If
DoEvents
Next i

Select Case formatnum
Case AVGFLUX
Case TIMEAVG, RMS
  Call subWriteHeater(setupin, Flux, OutputFileNumber, heaterlist)
  Call subWriteOffset(setupin, Flux, frmConvert!txtoffsetwrite.text, heaterlist)
End Select

'* close files
Close inputfilenumber
Close OutputFileNumber

If formatnum = MPEG Then
  Call MPEGWrite(OutputFileNumber, MaxPower, numscans)
End If
End Sub

Sub CalcFlux(OutData() As Double, ByRef Flux() As Double, offset() As Double, _
  ResFile1 As String, ResFile2 As String, AreaFile As String, _
  Temp As Double, numheaters As Integer, ByRef Reload As Integer)

```

```

Static ResSetup1 As CalSetupType, ResSetup2 As CalSetupType, AreaSetup As CalSetupType
Static Res1 As CalType, Res2 As CalType, Area As CalType
Dim ResFileNumber1 As Integer, ResFileNumber2 As Integer, _
    AreaFileNumber As Integer, i As Integer
'* Check if Reload is true
If Reload = True Then
    Reload = False
    '* Open Calibration files containing resistance
    '* data, and containing heater area data.
    ResFileNumber1 = FreeFile
    '* open file for input
    Open ResFile1 For Input As ResFileNumber1
    '* input setup data
    ResFileNumber2 = FreeFile
    Open ResFile2 For Input As ResFileNumber2
    AreaFileNumber = FreeFile
    Open AreaFile For Input As AreaFileNumber

    Call ReadCalSetup(ResSetup1, ResFileNumber1)
    Call ReadCalSetup(ResSetup2, ResFileNumber2)
    Call ReadCalSetup(AreaSetup, AreaFileNumber)
    '* Read resistance data
    Call ReadCal(Res1, ResFileNumber1)
    Call ReadCal(Res2, ResFileNumber2)
    Call ReadCal(Area, AreaFileNumber)
    '* Close Resistance Files
    Close ResFileNumber1
    Close ResFileNumber2
    Close AreaFileNumber
End If

'* Calculate Heat Flux in Watts/cm^2
For i = 0 To numheaters - 1
    '* Start with voltage value. Square it. Divide by the
    '* interpolated resistance value to get Watts. Divide by area
    '* to get watts/cm^2. subtract the offset to get rid of the
    '* substrate conduction heat transfer.
    Flux(i) = OutData(i) ^ 2 / (Temp * Res1.Vcmd(i) + Res2.Vcmd(i)) _
        / (Area.Vcmd(i) * 0.0001) - offset(i)
Next i

End Sub

Sub OpenMatrixFile(filename As String, OutputFileNumber As Integer, i As Integer)
    Dim appendage As String
    On Error Resume Next
    appendage = Trim(Str(i))
    If i = 0 Then appendage = "0"
    '* add zeros to fill out to 3 places
    appendage = String(3 - Len(appendage), "0") + appendage
    Open ChgExt(filename, appendage + ".dat") _
        For Output As OutputFileNumber
    If Err.Number = 53 Then
        MsgBox (Err.Description)
        Close OutputFileNumber
        Exit Sub
    End If
End Sub

End Sub

Sub ITWrite(OutputFileNumber)
    Dim x As Integer, y As Integer, a As Integer, b As Integer, _
        bMax As Integer
    Dim Values(160) As Double, itout(160) As Double
    For y = 7 To 0 Step -1
        itout(0) = Values(Position(0, y) - 1) ^ 2 / 10 * 255
        itout(1) = itout(0)
        itout(42) = Values(Position(7, y) - 1) ^ 2 / 10 * 255
        itout(43) = itout(42)
    For x = 0 To 7

```

```

        For a = 0 To 4
            itout(2 + 5 * x + a) = Values(Position(x, y) - 1) ^ 2 / 100 _
                * (255)
        Next a
    Next x
    Select Case y
        Case 0, 7
            bMax = 6
        Case 1 To 6
            bMax = 4
    End Select
    For b = 0 To bMax
        For a = 0 To 43
            Put OutputFileNumber, , itout(a)
        Next a
    Next b
Next y

End Sub

Sub MPEGWrite(OutputFileNumber, MaxPower, numscans As Integer)
    Dim BaseName As String
    Dim i As Integer
    '* write batch files necessary for
    '* making the MPEG movie

    '* open gnuplot script file
    BaseName = ChgExt(frmConvert!txtoutputfile.text, "")
    OutputFileNumber = FreeFile
    On Error Resume Next
    Open BaseName + ".gpt" For Output As OutputFileNumber
    If Err.Number = 53 Then
        MsgBox (Err.Description)
        Close OutputFileNumber
        Exit Sub
    End If

    Print #OutputFileNumber, "set key 8,0,0"
    Print #OutputFileNumber, "set cntrparam levels incr " _
        + Str(-MaxPower), Str(-MaxPower * 6 / 7)
    Print #OutputFileNumber, "set cntrparam levels 8"
    Print #OutputFileNumber, "set view 180, 0, 1, 1"
    Print #OutputFileNumber, "set nosurface;set contour base"
    Print #OutputFileNumber, "set cntrparam bspline"
    Print #OutputFileNumber, "set size 0.5,0.7"
    Print #OutputFileNumber, "n=0"
    Print #OutputFileNumber, "splot " + Chr(34) + BaseName _
        + ".dat" + Chr(34) + " index n title " _
        + Chr(34) + Chr(34) + " with lines"
    Print #OutputFileNumber, "set term postscript portrait color solid"
    Print #OutputFileNumber, "set output " + Chr(34) + BaseName _
        + ".ps" + Chr(34)
    For i = 0 To numscans - 1
        Print #OutputFileNumber, "n = " + Str(i) + " ; replot "

    Next i

    Close OutputFileNumber

    '* write batch file
    OutputFileNumber = FreeFile
    On Error Resume Next
    Open BaseName + ".bat" For Output As OutputFileNumber
    If Err.Number = 53 Then
        MsgBox (Err.Description)
        Close OutputFileNumber
        Exit Sub
    End If

    Print #OutputFileNumber, "gs -sOutputFile=" + BaseName _

```

```

    + "%d.pcx -sDEVICE=pcx256 -g220x200 -r35x40 " _
    + BaseName + ".ps"
For i = 1 To numscans
    Print #OutputFileNumber, "pcxtoppm < " + BaseName _
    + Trim(Str(i)) + ".pcx | pnmcut 60 10 160 120 > " + BaseName _
    + Trim(Str(i)) + ".ppm"
    Print #OutputFileNumber, "ppm2cyuv " + BaseName + Trim(Str(i)) _
    + ".ppm " + BaseName + Trim(Str(i)) + " -CCIR601"

Next i

Print #OutputFileNumber, "pvrgmpeg -XING -a 1 -b " _
+ Str(numscans) + " 3_5g -s " + BaseName + ".mpg"

Close OutputFileNumber

End Sub
Sub openinputfile(ByRef setupin As SetupType, ByRef inputfilenumber As Integer, filename As
String)
    '* open an input file for the "Convert" routine

    Dim datapoint As Integer
    inputfilenumber = FreeFile
    On Error Resume Next
    Open ChgExt(filename, ".TAG") For Input As inputfilenumber
    If Err.Number Then
        MsgBox (Err.Description)
        Close inputfilenumber
        Exit Sub
    End If
    Call ReadSetup(setupin, inputfilenumber)
    Close inputfilenumber
    '*input length is the number of heaters * 2 since we're inputing integers (2 bytes)
    Open filename For Binary As inputfilenumber Len = Len(datapoint) * 96

End Sub

Sub OpenOutputFile(formatnum As Integer, ByRef OutputFileNumber As Integer, filename As String)
    '* open the outputfile for "Convert"
    OutputFileNumber = FreeFile

    Select Case formatnum
    Case STANDARD, MATRICES, GNUPLOT, MPEG, MATRIXFILES, AVGFLUX, TIMEAVG, RMS, PROBDIST, FFT,
SAMPLE
        On Error Resume Next
        Open filename For Output As OutputFileNumber
        If Err.Number = 53 Then
            MsgBox (Err.Description)
            Close OutputFileNumber
            Exit Sub
        End If
    Case IMAGETOOL, BOIL
        On Error Resume Next
        Open filename For Binary As OutputFileNumber
        If Err.Number = 53 Then
            MsgBox (Err.Description)
            Close OutputFileNumber
            Exit Sub
        End If
    End Select

End Sub

Sub subTimeAvg(ByRef OutData() As Double, InData() As Double, numscans As Integer, numheaters As
Integer)
    Static Sum(160) As Double
    Static n As Integer
    Dim i As Integer

```

```

'* this subroutine depends on VB setting all declared variables = 0
'* when the program starts. Other languages don't do that.
'* sum heat fluxes for each heater
For i = 0 To numheaters
    Sum(i) = Sum(i) + InData(i)
Next i
'* increment the scan count
n = n + 1
'* check to see if all the scans have been summed
If n = numscans Then
    '* reset scan count
    n = 0
    '* set outdata equal to the average
    For i = 0 To numheaters - 1
        OutData(i) = Sum(i) / numscans
        '* reset sum to zero
        Sum(i) = 0
    Next i
End If
End Sub

Sub subUpdateStatus(numscans, i)
    Static oldPC As Long, PercentComplete As Long
    '* Display the percent complete
    oldPC = PercentComplete
    PercentComplete = CLng(i) * 10 / (numscans - 1)
    '* only refresh if it has changed by 10%.
    '* keeps the display from slowing down
    '* the program.

    If Not oldPC = PercentComplete Then
        frmConvert!txtStatus.text = Str$(PercentComplete * 10) + "% Complete"
        frmConvert!txtStatus.Refresh
        frmConvert!txtinputfile.Refresh
        frmConvert!txtoutputfile.Refresh
        frmConvert!txtavgflux.Refresh
        frmConvert!txtOffset.Refresh
    End If
End Sub

Sub subWriteHeater(setupin As SetupType, Flux() As Double, OutputFileNumber As Integer,
heaterlist() As Integer)
    Dim j As Integer, x As Integer, y As Integer
    Dim Values(160) As Double
    Dim ScanLine As String, sep As String
    '* outdata array doesn't necessarily contain the
    '* heater information in the order of the heater numbers.
    '* therefore we assign the Values array by remapping using
    '* the heaterlist array.
    sep = Chr$(9)
    For j = 0 To setupin.Heaters.Num - 1
        Values(heaterlist(j)) = Flux(j)
    Next j

    '* remap into array shaped like the heater.
    '* set heater 96 (97 starting at 1) to zero.
    Values(96) = 0

    For y = 8 To -1 Step -1
        For x = -1 To 8
            '* subtract 1 from "position" result because it starts at
            '* heater 1, not heater 0
            ScanLine = ScanLine + Format(Values(Position(x, y) - 1), "scientific") + sep
        Next x
        '* append a line feed
        ScanLine = ScanLine + CRLF
    Next y
    '* append a line feed
    ScanLine = ScanLine + CRLF

```

```

    '* output a dataset
    Print #OutputFileNumber, ScanLine

End Sub

Sub subWriteOffset(setupin As SetupType, Flux() As Double, offsetfilename As String, heaterlist()
As Integer)
    Dim freefilenum As Integer, i As Integer
    Dim CalSetup As CalSetupType
    Dim offset As CalType
    freefilenum = FreeFile
    Open offsetfilename For Output As freefilenum
    CalSetup.Comments = setupin.Comments
    CalSetup.Temp = setupin.Temp
    CalSetup.Heaters = setupin.Heaters
    offset.Comments = setupin.Comments
    offset.Temp = setupin.Temp

    For i = 0 To setupin.Heaters.Num - 1
        offset.Vcmd(i) = Flux(i)
    Next i
    Call WriteCalSetup(CalSetup, freefilenum)
    Call WriteCal(offset, freefilenum)
    Close freefilenum
End Sub

Sub subThresholdSample()
    Dim k(4) As Integer, inputfilenumber As Integer, OutputFileNumber As Integer, _
        i As Integer, numscans As Integer, Reload As Integer, SampleHeater As Integer

    Dim BoilFunc(1) As Byte
    Dim ResFile1 As String, ResFile2 As String, AreaFile As String, _
        sep As String
    Dim FluxThreshold As Double, dqdtthreshold As Double, _
        d2qdt2threshold As Double, offset(100) As Double, FullScale As Double, _
        TimeStep As Double, OutData(100) As Double, dqdt(1) As Double, _
        d2qdt2(1) As Double, time As Double

    Dim setupin As SetupType, setuptemp As SetupType
    Dim Flux(4) As fluxtype
    time = 0
    '* insure that resistance files will be loaded the first time through
    Reload = True
    ResFile1 = frmConvert!txtresfile1.text
    ResFile2 = frmConvert!txtresfile2.text
    AreaFile = frmConvert!txtAreaFile.text
    FluxThreshold = Val(frmConvert!txtfluxthreshold.text)
    dqdtthreshold = Val(frmConvert!txtdqdtThreshold.text)
    d2qdt2threshold = Val(frmConvert!txtd2qdt2Threshold.text)
    SampleHeater = frmConvert!txtThreshSamp.text

    '* define column seperator - chr$(9) is a Tab
    sep = Chr$(9)
    '* open input file and load setup information
    Call openinputfile(setupin, inputfilenumber, frmConvert!txtinputfile.text)
    Call OpenOutputFile(formatnum, OutputFileNumber, frmConvert!txtoutputfile.text)
    Call subReadOffset(offset, frmConvert!txtoffsetfile.text)
    Call subHeaterList(setupin)
    FullScale = SetFullScale(setupin.ADChwtype)

    TimeStep = 1 / setupin.SampRate
    numscans = setupin.Duration * setupin.SampRate

    '* set the initial value of the index variables
    k(0) = 0
    k(1) = 1

```



```

k(2) = 2

For i = 0 To numscans - 1
    time = TimeStep * i
    '* load voltage on heaters for one timestep into
    '* OutData matrix
    Call subLoadStep(setupin, inputfilenumber, FullScale, OutData)
    '* calculate power from voltage using resistance files
    '* place result in Flux matrix

    OutData(0) = OutData(SampleHeater)
    setuptemp = setupin
    setupin.Heaters.Num = 1

    Call CalcFlux(OutData, Flux(k(2)).Value, offset, ResFile1, ResFile2, AreaFile, _
        setupin.Temp, setupin.Heaters.Num, Reload)
    Call CalcBoilFunc(BoilFunc, setupin, Flux, k, TimeStep, _
        FluxThreshold, dqdtthreshold, d2qdt2threshold, dqdt, d2qdt2)
    Print #OutputFileNumber, Str(time) + sep _
        + Format(Flux(k(1)).Value(0), "scientific") + sep _
        + Format(dqdt(0), "scientific") _
        + sep + Format(d2qdt2(0), "scientific") _
        + sep + Str(BoilFunc(0) * 5)

    '* rotate the index variable
    k(3) = k(0)
    k(0) = k(1)
    k(1) = k(2)
    k(2) = k(3)

    setupin = setuptemp
    Call subUpdateStatus(numscans, i)
    DoEvents
Next i
Close OutputFileNumber
Close inputfilenumber

```

End Sub

### **G.1.4. Listing of FRMAUTOM.FRM**

```

Option Explicit
'* index to the sequence currently being specified
Dim SeqIndex As Integer
'* contains automation sequence data
Dim NewAuto(16) As SetupType
Dim AutoOut(16) As SetupType
Dim AutoIn(16) As SetupType

Private Sub chkActive_Click()
    If chkActive.Value = Unchecked Then
        NewAuto(SeqIndex).active = False
    Else
        NewAuto(SeqIndex).active = True
    End If
End Sub

Private Sub cmbfileformat_Click()
    NewAuto(SeqIndex).DataFormat = cmbfileformat.ListIndex
End Sub

```

```

Private Sub cmbGain_Click()
    NewAuto(SeqIndex).Gainindex = GainList.Value(cmbGain.ListIndex)
End Sub

Private Sub cmbSequence_Click()
    '* Change the index of the sequence that is
    '* being modified
    SeqIndex = cmbSequence.ListIndex
    '* update the screen
    Call Init
End Sub

Private Sub cmbTrigger_Click()
    NewAuto(SeqIndex).TrigSrc = TriggerList.Value(cmbTrigger.ListIndex)
End Sub

Private Sub cmdAdd_Click()
    Dim heater As Integer
    '* convert text to integer value
    heater = Val(txtHeaters)
    '* check if heaters is already selected
    If Not NewAuto(SeqIndex).Heaters.List(heater) = True Then
        '* set heater ON in the heater array
        NewAuto(SeqIndex).Heaters.List(heater) = True
        '* increment number of active heaters
        NewAuto(SeqIndex).Heaters.Num _
            = NewAuto(SeqIndex).Heaters.Num + 1
        '* add new
        lstHeaters.AddItem Str$(heater)
    End If
    If lstHeaters.ListIndex >= lstHeaters.ListCount _
        Or lstHeaters.ListIndex < 0 Then
        lstHeaters.ListIndex = 0
    End If
End Sub

Private Sub cmdApply_Click()
    Dim i
    For i = 0 To 15
        AutoSetup(i) = NewAuto(i)
    Next i
End Sub

Private Sub cmdBrowse_Click()
    frmCalFile!cmn1.DefaultExt = ".BIN"
    frmCalFile!cmn1.DialogTitle = "Data File Name"
    frmCalFile!cmn1.FILTER = "Data Files (*.BIN)|*.BIN|All Files (*.*)|*.*"
    frmCalFile!cmn1.Flags = cdIOFNHideReadOnly + cdIOFNOverwritePrompt + cdIOFNPathMustExist
    On Error Resume Next
    frmCalFile!cmn1.ShowSave
    If Not Err.Number = cdICancel Then
        txtFileName = frmCalFile!cmn1.FileName
        Call txtFileName_Change
    End If
End Sub

Private Sub cmdCancel_Click()
    Unload frmAutomation
End Sub

Private Sub cmdDefaults_Click()
    NewAuto(SeqIndex) = Defaults
    Call Init
End Sub

```

```

Private Sub cmdDelete_Click()
    If lstHeaters.ListCount > 0 Then
        Dim heater As Integer
        Dim Index As Integer
        '* assign the deleted heater to the text box
        '* so it could be quickly re-added
        Index = lstHeaters.ListIndex
        txtHeaters = lstHeaters.List(Index)
        heater = Val(lstHeaters.List(Index))
        '* remove heater from setup array
        NewAuto(SeqIndex).Heaters.List(heater) = False
        '* decrement number of active heaters
        NewAuto(SeqIndex).Heaters.Num _
            = NewAuto(SeqIndex).Heaters.Num - 1
        '* remove the heater from the list box
        lstHeaters.RemoveItem Index
        If Index < NewAuto(SeqIndex).Heaters.Num Then
            lstHeaters.ListIndex = Index
        ElseIf NewAuto(SeqIndex).Heaters.Num > 0 Then
            lstHeaters.ListIndex = Index - 1
        End If
    End If
End Sub

Private Sub cmdLoad_Click()
    Dim SetupFileNumber As Integer
    Dim SetupFileName As String
    Dim i As Integer
    '* set up properties of file box
    frmCalFile!cmn1.DefaultExt = ".aut"
    frmCalFile!cmn1.DialogTitle = "Load Setup File"
    frmCalFile!cmn1.FILTER = "Setup Files (*.AUT)|*.AUT|All Files (*.*)|*.*"
    frmCalFile!cmn1.Flags = cdlOFNHideReadOnly + cdlOFNPathMustExist + cdlOFNFileMustExist
    On Error Resume Next
    frmCalFile!cmn1.ShowOpen
    If Not Err.Number = cdlCancel Then
        SetupFileName = frmCalFile!cmn1.FileName
        SetupFileNumber = FreeFile
        On Error Resume Next
        Open SetupFileName For Input As #SetupFileNumber
        If Err Then MsgBox Err.Description
        Call ReadAuto(NewAuto, SetupFileNumber)
        Close SetupFileNumber
        Call Init
    End If
End Sub

Private Sub cmdOk_Click()
    Dim i
    For i = 0 To 15
        AutoSetup(i) = NewAuto(i)
    Next i

    Unload frmAutomation
End Sub

Private Sub cmdSave_Click()
    Dim SetupFileNumber As Integer
    Dim SetupFileName As String
    Dim i As Integer
    '* Initialize CommonDialogBox
    frmCalFile!cmn1.DefaultExt = ".AUT"
    frmCalFile!cmn1.DialogTitle = "Save Setup"
    frmCalFile!cmn1.FILTER = "Setup Files (*.AUT)|*.AUT|All Files (*.*)|*.*"
    frmCalFile!cmn1.Flags = cdlOFNHideReadOnly + cdlOFNOverwritePrompt + cdlOFNPathMustExist
    On Error Resume Next
    frmCalFile!cmn1.ShowSave
    If Not Err.Number = cdlCancel Then
        SetupFileName = frmCalFile!cmn1.FileName
    End If
End Sub

```

```

        SetupFileName = FreeFile
        Open SetupFileName For Output As #SetupFileName
        Call WriteAuto(NewAuto, SetupFileName)
        Close SetupFileName
    End If

End Sub

Private Sub Form_Load()
    Dim i As Integer
    SeqIndex = 0

    '* initialize combo box for gain control
    For i = 0 To GainList.Num - 1
        cmbGain.AddItem GainList.Text(i), i
    Next i
    '* initialize trigger options
    For i = 0 To TriggerList.Num - 1
        cmbTrigger.AddItem TriggerList.Text(i), i
    Next i
    '* initialize file format options
    For i = 0 To FormatList.Num - 1
        cmbfileformat.AddItem FormatList.Text(i), i
    Next i

    '* add 16 sequence numbers to the sequence box
    For i = 1 To 16
        cmbSequence.AddItem Str(i)
    Next i
    cmbSequence.ListIndex = 0

    For i = 0 To 15
        NewAuto(i) = AutoSetup(i)
    Next i
    '* Initialize Form Variables
    Call Init
End Sub

Sub AllEnable()
    NewAuto(SeqIndex).Heaters.method = ALLACTIVE
    txtRange(0).Enabled = False
    txtRange(1).Enabled = False
    txtHeaters.Enabled = False
    lstHeaters.Enabled = False
    cmdAdd.Enabled = False
    cmdDelete.Enabled = False
    optMethod(0).Value = True
    optMethod(1).Value = False
    optMethod(2).Value = False
End Sub

Sub Init()
    Dim i As Integer
    If NewAuto(SeqIndex).active = True Then
        chkActive.Value = Checked
    Else
        chkActive.Value = Unchecked
    End If
    cmbGain.ListIndex = Str(NewAuto(SeqIndex).Gainindex)
    txtSampRate.Text = Str(NewAuto(SeqIndex).SampRate)
    txtDuration.Text = Str(NewAuto(SeqIndex).Duration)
    txtDelayTime.Text = Str(NewAuto(SeqIndex).DelayTime)
    cmbTrigger.ListIndex = NewAuto(SeqIndex).TrigSrc
    txtTemp.Text = Str(NewAuto(SeqIndex).Temp)

    txtRange(0).Text = Str(NewAuto(SeqIndex).Heaters.Range(0))
    txtRange(1).Text = Str(NewAuto(SeqIndex).Heaters.Range(1))
    Do While lstHeaters.ListCount > 0
        lstHeaters.RemoveItem 0
    
```

```

Loop

For i = 0 To ADCLASTCHANNEL
    If NewAuto(SeqIndex).Heaters.List(i) = True Then
        lstHeaters.AddItem Str$(i)
    End If
Next i
If lstHeaters.ListCount > 0 Then
    lstHeaters.ListIndex = 0
End If

If NewAuto(SeqIndex).Heaters.method = ALLACTIVE Then
    Call AllEnable
ElseIf NewAuto(SeqIndex).Heaters.method = RANGEACTIVE Then
    Call RangeEnable
ElseIf NewAuto(SeqIndex).Heaters.method = SPECIFIC Then
    Call SpecificEnable
End If
txtHeaters.Text = ""
txtComments.Text = NewAuto(SeqIndex).Comments
txtFileName.Text = NewAuto(SeqIndex).FileName
cmbfileformat.ListIndex = NewAuto(SeqIndex).DataFormat

End Sub

Sub RangeEnable()
    NewAuto(SeqIndex).Heaters.method = RANGEACTIVE
    txtRange(0).Enabled = True
    txtRange(1).Enabled = True
    txtHeaters.Enabled = False
    lstHeaters.Enabled = False
    cmdAdd.Enabled = False
    cmdDelete.Enabled = False
    optMethod(0).Value = False
    optMethod(1).Value = True
    optMethod(2).Value = False

End Sub

Sub SpecificEnable()
    NewAuto(SeqIndex).Heaters.method = SPECIFIC
    txtRange(0).Enabled = False
    txtRange(1).Enabled = False
    txtHeaters.Enabled = True
    lstHeaters.Enabled = True
    cmdAdd.Enabled = True
    cmdDelete.Enabled = True
    optMethod(0).Value = False
    optMethod(1).Value = False
    optMethod(2).Value = True

End Sub

Private Sub optMethod_Click(Index As Integer)
    If Index = 0 Then
        Call AllEnable
    ElseIf Index = 1 Then
        Call RangeEnable
    ElseIf Index = 2 Then
        Call SpecificEnable
    End If
End Sub

Private Sub optMethod_KeyPress(Index As Integer, KeyAscii As Integer)
    Dim newIndex As Integer
    If Index < 2 Then

```

```

        newindex = Index + 1
    Else
        newindex = 0
    End If
    optMethod(newindex).SetFocus
End Sub

Private Sub txtComments_Change()
    NewAuto(SeqIndex).Comments = txtComments.Text
End Sub

Private Sub txtDelayTime_Change()
    NewAuto(SeqIndex).DelayTime = Val(txtDelayTime.Text)
End Sub

Private Sub txtDelayTime_LostFocus()
    txtDelayTime.Text = Str(NewAuto(SeqIndex).DelayTime)
End Sub

Private Sub txtDuration_Change()
    NewAuto(SeqIndex).Duration = Val(txtDuration.Text)
End Sub

Private Sub txtDuration_LostFocus()
    txtDuration.Text = Str(NewAuto(SeqIndex).Duration)
End Sub

Private Sub txtFileName_Change()
    NewAuto(SeqIndex).FileName = txtFileName.Text
End Sub

Private Sub txtRange_Change(Index As Integer)
    Dim limit As Integer
    '* heater range is specified from 0 to adclastchannel
    limit = Val(txtRange(Index))
    If limit > -1 And limit < ADCLASTCHANNEL + 1 Then
        NewAuto(SeqIndex).Heaters.Range(Index) = limit
    End If
    txtRange(Index).Text =
        = Str$(NewAuto(SeqIndex).Heaters.Range(Index))
End Sub

Private Sub txtSampRate_Change()
    NewAuto(SeqIndex).SampRate = Val(txtSampRate.Text)
End Sub

Private Sub txtSampRate_LostFocus()
    txtSampRate.Text = Str(NewAuto(SeqIndex).SampRate)
End Sub

Private Sub txtTemp_Change()
    NewAuto(SeqIndex).Temp = Val(txtTemp.Text)
End Sub

Private Sub txtTemp_LostFocus()
    txtTemp.Text = Str(NewAuto(SeqIndex).Temp)
End Sub

```

### **G.1.5. Listing of FRMCONFL.FRM**

```
Option Explicit
Dim answer As Integer

Public Function yesno(Caption As String) As Integer
    *****
    '* returns True if user clicks on "yes" and
    '* False if user clicks on "no"
    *****
    lblConfirm = Caption
    yesno = False
    answer = False
    Show 1
    yesno = answer
End Function

Private Sub cmdNo_Click()
    Unload frmConfirm
End Sub

Private Sub cmdYes_Click()
    answer = True
    Unload frmConfirm
End Sub
```

### **G.1.6. Listing of FRMVIEWC.FRM**

```
Option Explicit

Sub RangeEnable()
    txtRange(0).Enabled = True
    txtRange(1).Enabled = True
    lstHeaters.Enabled = False
    optMethod(0).Value = False
    optMethod(1).Value = True
    optMethod(2).Value = False
End Sub

Sub SpecificEnable()
    txtRange(0).Enabled = False
    txtRange(1).Enabled = False
    lstHeaters.Enabled = True
    optMethod(0).Value = False
    optMethod(1).Value = False
    optMethod(2).Value = True
End Sub

Sub AllEnable()
    txtRange(0).Enabled = False
    txtRange(1).Enabled = False
    lstHeaters.Enabled = False
    optMethod(0).Value = True
    optMethod(1).Value = False
    optMethod(2).Value = False
End Sub
```

### **G.1.7. Listing of FRMDAC.FRM**

```
Option Explicit
*****
```

```

'* This form exists to handle comm port activities
'* using the MSComm control
'*****

Private Sub comDAC_OnComm()
    'Beep
    If comdac.CommEvent = comEvDSR Then
        '* Don't do anything - DSR changed from
        '* TRUE to FALSE meaning
    Else

        MsgBox "comDAC.CommEvent = " + Str$(comdac.CommEvent)
    End If
End Sub

Private Sub Form_Load()
    '*****
    '* NOTE: If you put a comDAC.PortOpen = false
    '* statement right after an Output method, the
    '* output will never get to the comm port,
    '* because the port will close too early.
    '*
    '* The computer control board requires a carriage
    '* return, line-feed combination, so I defined
    '* a variable in modDAC called CRLF to append
    '* to all output.
    '*****

End Sub

```

### **G.1.8. Listing of FRMADC.FRM**

```

Option Explicit

Private Sub tmrADC_Timer()
    Call ADCTimerSub
End Sub

Private Sub tmrWait_Timer()
    Dim done As Integer, retval As Integer
    '* Check to see if the A/D boards are finished
    '* collecting data. If so, then read the data
    '* and set controls so heaters, temperature, etc.
    '* can be reset.

    retval = cbDBitIn(0, FIRSTPORTA, 20, done)
    If done = 1 Then Call ADCReadCustom
End Sub

```

### **G.1.9. Listing of FRMAUTO.FRM**

```

Option Explicit
Const INDEXNOFILL = 0
Const INDEXGETDATA = 0
Const INDEXMANUAL = 0

Sub GridInit()
    Dim i As Integer

```



```

grdAuto.FixedRows = 1
grdAuto.FixedCols = 1
grdAuto.Rows = 16
grdAuto.Cols = 5
grdAuto.ColWidth(0) = grdAuto.Width / grdAuto.Cols

For i = 1 To 4
    grdAuto.ColWidth(i) = grdAuto.Width / grdAuto.Cols - 110
Next i
grdAuto.RowHeight(0) = (grdAuto.Height / 14 - 10) * 2
For i = 1 To 15
    grdAuto.RowHeight(i) = grdAuto.Height / 14 - 10
Next i

grdAuto.col = 0
For i = 1 To 15
    grdAuto.Row = i
    grdAuto.text = Str(i)
Next i
grdAuto.Row = 0
grdAuto.col = 1
grdAuto.text = "Temperature"
grdAuto.col = 2
grdAuto.text = "Input File"
grdAuto.col = 3
grdAuto.text = "DaqBook File"
grdAuto.col = 4
grdAuto.text = "High-speed file"

grdAuto.Row = 1
grdAuto.col = 1

End Sub

Private Sub cmbNumFormat_Click()
    cmbNumFormat.ListIndex = INDEXNOFILL
End Sub

Private Sub cmbTask_Click()
    cmbTask.ListIndex = INDEXGETDATA
End Sub

Private Sub cmbTempMode_Click()
    cmbTempMode.ListIndex = INDEXMANUAL
End Sub

Private Sub cmdCopy_Click()
    Dim CellText As String
    Dim i As Integer, j As Integer, OldRow As Integer
    OldRow = grdAuto.Row
    CellText = grdAuto.text
    For i = 1 To grdAuto.Rows - 1
        grdAuto.Row = i
        grdAuto.text = CellText
    Next i
    grdAuto.Row = OldRow
End Sub

Private Sub cmdDelete_Click()
    Dim i As Integer, j As Integer, OldRow As Integer, OldCol As Integer
    OldRow = grdAuto.Row
    OldCol = grdAuto.col
    For i = grdAuto.SelStartRow To grdAuto.SelEndRow
        For j = grdAuto.SelStartCol To grdAuto.SelEndCol
            grdAuto.Row = i
            grdAuto.col = j
            grdAuto.text = ""
        Next j
    Next i
    grdAuto.Row = OldRow

```

```

    grdAuto.col = OldCol
End Sub

Private Sub cmdLoadAuto_Click()
    Dim FileNumber As Integer, i As Integer, j As Integer
    Dim junk As String
    FileNumber = FreeFile
    Open txtAutoFileName.text For Input As FileNumber
    Input #FileNumber, junk
    txtDelayTime.text = junk

    For i = 1 To grdAuto.Cols - 1
        For j = 1 To grdAuto.Rows - 1
            grdAuto.Row = j
            grdAuto.col = i
            Input #FileNumber, junk
            grdAuto.text = junk
        Next j
    Next i
    Close FileNumber
End Sub

Private Sub cmdSave_Click()
    Dim FileNumber As Integer, i As Integer, j As Integer
    FileNumber = FreeFile
    Open txtAutoFileName.text For Output As FileNumber
    Write #FileNumber, txtDelayTime.text
    For i = 1 To grdAuto.Cols - 1
        For j = 1 To grdAuto.Rows - 1
            grdAuto.Row = j
            grdAuto.col = i
            Write #FileNumber, grdAuto.text
        Next j
    Next i
    Close FileNumber
End Sub

Private Sub cmdStartAuto_Click()
    grdAuto.Row = 1
    grdAuto.col = 1
    frmMain.txtTemp = grdAuto.text

    tmrAuto.Enabled = True
End Sub

Private Sub Form_Load()
    GridInit
    txtChangeGrid.Top = grdAuto.Top - txtChangeGrid.Height
    lblChangeGrid.Height = txtChangeGrid.Height
    lblChangeGrid.Top = txtChangeGrid.Top
    lblChangeGrid.Caption = "Edit Cell:"
    txtBaseName.text = ""
    txtFirstNum.text = ""
    txtLastNum.text = ""
    cmbNumFormat.AddItem "Don't Fill"
    cmbNumFormat.AddItem "Fill with zeros"
    cmbNumFormat.ListIndex = 0
    txtNumLength = ""
    cmbTempMode.AddItem "Manual Entry"
    cmbTempMode.AddItem "Range"
    cmbTempMode.AddItem "Randomize"
    cmbTempMode.ListIndex = 0
    txtLowTemp.text = ""
    txtHighTemp.text = ""
    txtTempInterval.text = ""
    cmbTask.AddItem "Data Acquisition"
    cmbTask.AddItem "Data Reduction"
    cmbTask.AddItem "Both"
    cmbTask.ListIndex = 0

```

```

    tmrAuto.Enabled = False
    tmrHighSpeed.Enabled = False
    tmrTemp.Enabled = False
    '* set two second interval, to make sure to include
    '* 1.6 second DAC time.
    tmrTemp.interval = 2000
    '* set Timer Intervals
    '* 1 minute interval
    tmrAuto.interval = 60000
    '* 1 second interval
    tmrHighSpeed.interval = 1000
    '* 2 second interval allows time for high-speed
    '* data-acquisitions to complete

End Sub

Private Sub grdAuto_Click()
    txtChangeGrid.SetFocus
End Sub

Private Sub grdAuto_RowColChange()
    txtChangeGrid.text = grdAuto.text
End Sub

Private Sub tmrAuto_Timer()
    Static Minutes As Integer
    Minutes = Minutes + 1
    If Minutes = Val(txtDelayTime.text) Then
        Minutes = 0
        frmMain!cmdStart.Value = True
        tmrHighSpeed.Enabled = True
        tmrAuto.Enabled = False
    End If
End Sub

Private Sub tmrHighSpeed_Timer()
    Static seconds As Integer
    seconds = seconds + 1
    '* wait until the data acquisition is complete
    If seconds = SetupData.Duration + 5 Then
        seconds = 0
        tmrHighSpeed.Enabled = False
        frmMain!cmdStart.Value = True
        tmrTemp.Enabled = True
    End If
End Sub

Private Sub tmrTemp_Timer()
    '* check if there are more datapoints to take
    '* if not, then turn off the timer and don't do anything else
    '* Otherwise, perform the following steps
    '* change temperature
    '* change Output File Name
    '* Enable Wait timer
    tmrAuto.Enabled = True
    tmrTemp.Enabled = False
End Sub

Private Sub txtAutoFileName_Click()
    Static filename As String
    frmCalFile!cmn1.filename = filename
    frmCalFile!cmn1.DefaultExt = ".AUT"
    frmCalFile!cmn1.DialogTitle = "Data File Name"
    frmCalFile!cmn1.FILTER = "Data Files (*.AUT)|*.AUT|All Files (*.*)|*.*"
    frmCalFile!cmn1.Flags = cdloFNHideReadOnly + cdloFNOverwritePrompt + cdloFNPathMustExist
    On Error Resume Next

```

```

    frmCalFile!cmn1.ShowSave
    If Not Err.Number = cdlCancel Then
        txtAutoFileName = frmCalFile!cmn1.filename
        filename = frmCalFile!cmn1.filename
    End If
End Sub

Private Sub txtChangeGrid_Change()
    grdAuto.text = txtChangeGrid
End Sub

Private Sub txtChangeGrid_GotFocus()
    txtChangeGrid.SelStart = 0
    txtChangeGrid.SelLength = Len(txtChangeGrid.text)
End Sub

```

### **G.1.10. Listing of FRMBATCH.FRM**

```

Dim SampleHeater(100) As Integer, numheaters(100) As Integer, numfiles As Integer, _
    NumFormat(100) As Integer
Dim inputfilename(100) As String, outputfilename(100) As String, _
    offsetfilename(100) As String, boilfilename(100) As String

Dim FluxThreshold(100) As Double, dqdtthreshold(100) As Double, d2qdt2threshold(100) As Double

Dim activeForm As Form

Private Sub cmdAutomate_Click()
    Dim i As Integer, j As Integer
    Dim Flux As Double
    Dim freefilenum As Integer
    Dim offset(119) As Double
    Dim strng As String
    Dim frmActive As Form
    Set frmActive = frmConvert
    Do While 1 + 1 = 3
        '* input a list of offset values from a file
        freefilenum = FreeFile
        Open "\trule\dat\sc.dat" For Input As freefilenum
        For i = 0 To 14
            Input #freefilenum, offset(i)
            '* assign 15 to 29 the descending order values
            offset(29 - i) = offset(i)
        Next i
        Close freefilenum

        '* assign the other values of offset in the same
        '* order as 0 to 29
        For i = 0 To 29
            offset(30 + i) = offset(i)
            offset(60 + i) = offset(i)
            offset(90 + i) = offset(i)
        Next i
    Loop

    freefilenum = FreeFile
    'Open "\trule\dat\062297\0622flux.dat" For Output As freefilenum
    Do While 1 + 1 = 3
        '* convert to rms and avgflux data
        frmActive!cmbFormat.ListIndex = 7

        For i = 0 To 59
            With frmActive
                .txtinputfile.text = "\trule\dat\062397\0623_" + Trim(Str(i)) + ".bin"
                .txtoutputfile.text = "\trule\dat\062397\0623_" + Trim(Str(i)) + ".avg"
                .txtOffset.text = Str(offset(i))
                .cmdConvert.Value = True
            End With
        Next i
    Loop

```

```

        'Print #freefilenum, .txtavgflux.Text
    End With
    Next i
    'Close freefilenum
Loop
'* Convert to RMS values

'* convert to rms and avgflux data
cmbFormat.ListIndex = 7

For i = 2 To 21
    strng = Trim(Str(i))
    If i < 10 Then strng = "0" + strng
    frmActive!txtinputfile.text = "\trule\dat\061797\0617_" + strng + "c.bin"
    frmActive!txtoutputfile.text = "\trule\dat\061797\0617_" + strng + "c.avg"
    'txtOffset.Text = Str(offset(i))
    frmActive!cmdConvert.Value = True
    'Print #freefilenum, txtavgflux.Text
Next i

Do While 1 + 1 = 3

    '* Redo the above conversion for the nucleate boiling curve data
    '* from 6/22
    For i = 0 To 6
        offset(13 - i) = offset(i)
    Next i
    '* assign the other values of offset in the same
    '* order as 0 to 14
    For i = 0 To 13
        offset(14 + i) = offset(i)
        offset(28 + i) = offset(i)
        offset(42 + i) = offset(i)
    Next i

    cmbFormat.ListIndex = 6

    freefilenum = FreeFile
    Open "\trule\dat\062397\0624a.flx" For Output As freefilenum

    For i = 0 To 59
        With frmActive
            txtinputfile.text = "\trule\dat\062497\0624_" + Trim(Str(i)) + "a.bin"
            txtoutputfile.text = "\trule\dat\062497\0624_" + Trim(Str(i)) + "a.dat"
            'txtOffset.Text = Str(offset(i))
            cmdConvert.Value = True
            Print #freefilenum, .txtavgflux.text
        End With
    Next i
    Close freefilenum
Loop
End Sub

Private Sub cmdAveraging_Click()
    Dim freefile1 As Integer, freefile2 As Integer, freefile3 As Integer, freefile4 As Integer
    Dim ncfilenum(100) As Integer, datafilenum(100) As Integer
    Dim scnc(100) As Double
    Dim ncfiletxt As String, datafiletxt As String
    Dim ncfilename As String, datafilename As String, outfilename As String
    Dim OutData As Double
    Dim txtline As String
    Dim scncnew As Double
    Dim scncnewtxt As String, datatxt As String
    Dim i As Integer, j As Integer, k As Integer
    Dim ncscnewtxt As String
    Dim data As Integer
    Dim sep As String
    Dim Sum As Double, avg As Double
    Dim frmActive As Form
    Set frmActive = frmConvert
    sep = Chr$(9)

```

```

freefile1 = FreeFile
Open "\trule\dat\scnc.dat" For Input As freefile1
For i = 0 To 14
    Input #freefile1, scnc(i)
    '* assign 15 to 29 the descending order values
    scnc(29 - i) = scnc(i)
    ncfilenum(i) = i + 7
    ncfilenum(29 - i) = i + 7
Next i
Close freefile1

For i = 0 To 29
    scnc(30 + i) = scnc(i)
    ncfilenum(30 + i) = ncfilenum(i)
Next i
freefile4 = FreeFile
Open "\trule\dat\062397\0623uncr.dat" For Output As freefile4

For i = 0 To 59

    ncfiletxt = Trim(Str(ncfilenum(i)))
    datafiletxt = Trim(Str(i))

    If ncfilenum(i) < 10 Then ncfiletxt = "0" + ncfiletxt
    ncfilename = "\trule\dat\061797\0617_" + ncfiletxt + "c.avg"
    datafilename = "\trule\dat\062397\0623_" + datafiletxt + ".avg"
    'outfilename = "\trule\dat\062397\0623_" + datafiletxt + "b.avg"
    freefile1 = FreeFile
    Open ncfilename For Input As freefile1
    freefile2 = FreeFile
    Open datafilename For Input As freefile2
    'freefile3 = FreeFile
    'Open outfile for Output As freefile3
    For j = 0 To 9
        For k = 0 To 9
            Input #freefile1, scncnew
            Input #freefile2, data
            OutData = data + scnc(i) ' - scncnew
            If j = 0 And k = 0 _
            Or j = 0 And k = 9 _
            Or j = 9 And k = 0 _
            Or j = 9 And k = 9 _
            Then OutData = 0

            Sum = Sum + OutData
            txtline = txtline + format(OutData, "scientific") + sep
        Next k
        'Print #freefile3, txtline
        txtline = ""
    Next j
    avg = Sum / 96
    Print #freefile4, format(avg, "scientific")
    Sum = 0

    Close freefile1
    Close freefile2
    'Close freefile3
Next i
Close freefile4

End Sub

Private Sub cmdBatch_Click()
    Dim inputfilenumber As Integer
    Dim blank As String
    batchfilenumber = FreeFile
    Open txtBatchFileName.text For Input As batchfilenumber
    Input #batchfilenumber, blank
    Input #batchfilenumber, numfiles

```

```

    '* Input an extra line to provide space for labels
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, NumFormat(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, inputfilename(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, outputfilename(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, offsetfilename(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, boilfilename(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, numheaters(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, FluxThreshold(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, dqdtthreshold(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, d2qdt2threshold(i)
    Next i
    Input #batchfilenumber, blank
    For i = 0 To numfiles - 1
        Input #batchfilenumber, SampleHeater(i)
    Next i

    tmrbatch.Enabled = True
End Sub

Private Sub Form_Load()
    tmrbatch.Enabled = False
    tmrbatch.interval = 1000
    Set activeForm = frmConvert
End Sub

Private Sub tmrbatch_Timer()
    Static i

    If activeForm!cmdConvert.Enabled = True Then
        activeForm!cmbFormat.ListIndex = NumFormat(i)
        activeForm!txtinputfile.text = inputfilename(i)
        activeForm!txtoutputfile.text = outputfilename(i)
        activeForm!txtoffsetfile.text = offsetfilename(i)
        activeForm!txtBoilFile.text = boilfilename(i)
        activeForm!txtNumHeaters.text = Str(numheaters(i))
        activeForm!txtfluxthreshold.text = Str(FluxThreshold(i))
        activeForm!txtdqdtThreshold.text = Str(dqdtthreshold(i))
        activeForm!txtd2qdt2Threshold.text = Str(d2qdt2threshold(i))
        activeForm!txtThreshSamp.text = Str(SampleHeater(i))
        activeForm!cmdConvert.Value = True
        i = i + 1
    End If
    If i = numfiles Then

```

```

        tmrbatch.Enabled = False
        i = 0
    End If
End Sub

```

## G.2. LISTING OF "CAL.VBP"

```

Form=FRMMAIN.FRM
Module=modDAC; ..\TRIM\MODDAC.BAS
Module=modADC; ..\TRIM\MODADC.BAS
Module=modShared; ..\CONTROL\MODSHARE.BAS
Module=modDagBook; ..\DAQBOOK.BAS
Module=modMain; MODMAIN.BAS
Form=..\TRIM\FRMCONFI.FRM
Form=FRMSETUP.FRM
Form=FRMCOMMO.FRM
Form=..\TRIM\FRMDAC.FRM
Form=..\CONTROL\FRMADC.FRM
Form=..\CONTROL\FRMCALFI.FRM
Module=modCBW; ..\..\..\CB\VBWIN\CBW.BAS
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMDLG16.OCX
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL16.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST16.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID16.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#1.0#0#C:\WINDOWS\SYSTEM\OC25.DLL#Standard OLE
Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#2.5#0#C:\WINDOWS\SYSTEM\DAO2516.DLL#Microsoft
DAO 2.5 Object Library
Object={648A5603-2C6E-101B-82B6-000000000014}#1.0#0; MSCOMM16.OCX
ProjWinSize=46,385,251,413
ProjWinShow=2
IconForm="frmMain"
HelpFile=""
ExeName="CAL.EXE"
Name="Calibration"
HelpContextID="0"
StartMode=0
VersionCompatible="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="University of Denver"

```

### G.2.1. Listing of MODDAC.BAS

```

Attribute VB_Name = "modDAC"
Option Explicit
'*****
' * This module contains code for declaring D/A
' * data types and handling low-level D/A functions. It
' * was designed for controlling the computer control board
' * for outputting Vcmd values.
'*****

'*****
' * The type DACType contains only the information
' * that the software cares about. The low-level
' * hardware information will be contained in other
' * structures and variables which the user and the
' * software don't care about.
'*****
Public Type DACType
'*****

```



```

    '* Vcmds values given in units of V.
    '* first index is heater number, second
    '* index is the number of the temperature
    '* table.
    '*****

    Vcmd(160, 16) As Single
    '* Indicates whether a table should
    '* be filled with values when DACPutData
    '* is called
    TableActive(16) As Boolean
End Type
    '* contains the ASCII values of a
    '* a carriage-return and a line-feed
Public CRLF As String
    '* the general variable containing the
    '* DAC information
Public dac As DACType
    '* Set communication port number
Public Const PortNum = 1
    '* Set to indicate whether the DAC hardware
    '* is connected or not
Public DACHardwareType As Integer
Public Const DACNOHARDWARE = 0
Public Const DACCCB = 1

Public Type DACListType
    Num As Integer
    text(5) As String
    Value(5) As Integer
End Type
Public DACList As DACListType

Sub Cal2DAC(CalData As CalType, TableNum As Integer)
    Dim i
    '* assign vcmd values from caldata to DAC
    For i = 0 To ADCLASTCHANNEL
        dac.Vcmd(i, TableNum) = CalData.Vcmd(i)
    Next i
    '* set table TableNum active
    dac.TableActive(TableNum) = True
End Sub

Sub DACClose()
    '* called when the program is exiting, typically
    '* close the comm port when this form is unloaded
    '* which always happen, and only happen, when
    '* the program exits.
    Select Case DACHardwareType
        Case DACCCB
            frmDAC!comdac.PortOpen = False
            Unload frmDAC
    End Select
End Sub

Sub DACDefine()
    '* initialize the DAC variables
    DACList.Num = 2
    DACList.Value(0) = DACNOHARDWARE
    DACList.Value(1) = DACCCB
    DACList.text(0) = "No Hardware"
    DACList.text(1) = "Comp. Control Board"
End Sub

Sub DACSetTable(j As Integer)
    '* Set which table in the hardware is being used to output

```

```

    '* Vcmd signals
    Select Case DACHardwareType
        Case DACCCB
            frmDAC!comdac.output = CRLF + "T" + Hex(j) _
                + CRLF
            dac.TableActive(j) = True
    End Select
End Sub

Function HexAddr(HeaterNum As Integer, TableNumber) As String

    '*****
    '* This subroutine converts a channel address
    '* into a hex string of the
    '* form required by the computer control board
    '*****

    '*****
    '* The binary address consists of 3 zeros,
    '* followed by 4 card select digits, followed
    '* by 4 channel select digits, followed by 1
    '* zero. The highest index is the most sig.
    '* digit (leftmost digit)
    '*****
    Dim CS As Integer
    Dim ChS As Integer
    Dim DecAddr As Integer
    Dim i As Integer
    Dim BinCS(4) As Integer, BinChS(4) As Integer
    Dim BinAddr(12) As Integer

    '* convert heater number to card select, channel
    '* select. HeaterNum from 0 to ADCLASTCHANNEL

    '* Card select from 0 to 15

    CS = HeaterNum \ 16
    '* Channel Select from 0 to 9
    ChS = HeaterNum Mod 16
    BinAddr(0) = 0
    BinAddr(9) = 0
    BinAddr(10) = 0
    BinAddr(11) = 0

    For i = 3 To 0 Step -1

        '* Convert Decimal Card Select into Binary
        '* Card Select

        If CS >= 2 ^ i Then
            CS = CS - 2 ^ i
            BinCS(i) = 1
        Else
            BinCS(i) = 0
        End If

        '* Convert decimal Channel Select into Binary
        '* Channel Select

        If ChS >= 2 ^ i Then
            ChS = ChS - 2 ^ i
            BinChS(i) = 1
        Else
            BinChS(i) = 0
        End If

        '* map Card select and channel select into
        '* binary address

        BinAddr(i + 1) = BinChS(i)
        BinAddr(i + 5) = BinCS(i)
    
```

```

Next i
For i = 0 To 11
    DecAddr = DecAddr + 2 ^ i * BinAddr(i)
Next i

'* add on &HC000 to get the base address
'* add on &H200 * tablenumber to get table address

DecAddr = DecAddr + &HC000 + &H200 * TableNumber
HexAddr = dectohex(DecAddr, 4)

End Function

Function dectohex(ByVal Decimal As Integer, Digits As Integer) As String
    Dim hex1 As String
    Dim LeadingZeros As Integer, i As Integer
    Dim length As Integer
    '* converts a decimal integer to a HEX string of the length
    '* Digits. Adds leading zero's to fill out to the correct
    '* length.

    '* convert vcmd to hex value
    hex1 = Hex$(Decimal)
    '* calculate number of leading zero for 3 digits
    '* total string length
    length = Len(hex1)
    LeadingZeros = Digits - Len(hex1)
    '* add leading zeros to string
    For i = 1 To LeadingZeros
        hex1 = "0" + hex1
    Next i
    dectohex = hex1
End Function

Sub DACInit()
    Dim i As Integer
    Dim junk As String

    Select Case DACHardwareType
        Case DACCCB
            CRLF = Chr$(13) + Chr$(10)

            '* load the form that has the COM port controls
            Load frmDAC

            '* Only run the following lines if hardware
            '* is hooked up
            frmDAC!comdac.CommPort = PortNum
            frmDAC!comdac.Settings = "9600,N,8,1"
            frmDAC!comdac.InputLen = 1
            frmDAC!comdac.PortOpen = True

            *****
            '* send a reset command, and set the calibration
            '* table to 0 volts
            *****

            frmDAC!comdac.output = CRLF + "X" + CRLF
            Do While frmDAC!comdac.InBufferCount = 0
                '* wait until something comes into the
                '* input buffer
            Loop
            Do While frmDAC!comdac.InBufferCount > 0
                '* input 1 character at a time until
                '* the input buffer is empty
                junk = frmDAC!comdac.Input
            Loop
            frmDAC!comdac.output = CRLF + "C0" + CRLF
            '* Set table to T0

```

```

        If frmDAC!comdac.DSRHolding = False Then frmDAC!comdac.output = CRLF + "T0" + CRLF

        For i = 0 To 15
            dac.TableActive(i) = False
        Next i
    End Select
End Sub

Public Sub DACPutData(DACval As DACType)

    '* Write Vcmd data to all the active tables on the
    '* microprocessor control card.

    Dim i As Integer, j As Integer
    Dim Address As String

    For j = 0 To 15
        If dac.TableActive(j) = True Then
            '* Step through 160 Vcmd Channels
            For i = 0 To ADCLASTCHANNEL
                '* dac.vcmd is given in mV
                Address = "S" _
                    + HexAddr(i, j) _
                    + dectohex(DACval.Vcmd(i, j) * &HFFF / 10000, 3) _
                    + "0" + CRLF
                Select Case DACHardwareType
                    Case DACCCB
                        frmDAC!comdac.output = Address
                        '* wait until output buffer is clear
                        Do While frmDAC!comdac.OutBufferCount
                            Loop
                End Select

                Next i
            End If
        End If
    Next j

End Sub

Sub SetHeater(HeaterNum As Integer, Voltage As Double)

    *****
    '* Output a string to Table zero of the controller board to
    '* change the resistance of one heater.
    *****
    '* voltage is in units of mV

    dac.Vcmd(HeaterNum, 0) = Voltage
    Dim Address As String
    Address = CRLF + "S" _
        + HexAddr(HeaterNum, 0) _
        + dectohex(Voltage * &HFFF / 10000, 3) _
        + "0" + CRLF
    ' frmADCStatus!lbl2.Caption = Address
    Select Case DACHardwareType
        Case DACCCB

            frmDAC!comdac.output = Address
    End Select

End Sub

```

### **G.2.2. Listing of MODMAIN.BAS**

```
Attribute VB_Name = "modMain"
Public SetupDone As Boolean
Public SetupData As CalSetupType
Public SetupOut As CalSetupType
Public SetupIn As CalSetupType
Public Cal As CalType
```

### **G.2.3. Listing of FRMSETUP.FRM**

```
Dim NewSetup As CalSetupType

Dim Default As CalSetupType

Private Sub Combo1_Change()

End Sub

Private Sub cmbADChwtype_Change()

End Sub

Private Sub cmbADChwtype_Click()
    NewSetup.ADChwtype = cmbADChwtype.ListIndex
End Sub

Private Sub cmbGain_Click()
    'commented out because I'm not using variable gain
    'NewSetup.Gainindex = cmbGain.ListIndex
End Sub

Private Sub cmdAdd_Click()
    Dim heater As Integer
    '* convert text to integer value
    heater = Val(txtHeaters.text)
    '* check if heaters is already selected
    If Not NewSetup.Heaters.List(heater) = True Then
        '* set heater ON in the heater array
        NewSetup.Heaters.List(heater) = True
        '* increment number of active heaters
        NewSetup.Heaters.Num = NewSetup.Heaters.Num + 1
        '* add new
        lstHeaters.AddItem Str$(heater)
    End If
    If lstHeaters.ListIndex >= lstHeaters.ListCount _
        Or lstHeaters.ListIndex < 0 Then
        lstHeaters.ListIndex = 0
    End If

End Sub

Private Sub cmdADSetup_Click()
    frmADSetup.Show 1

End Sub

Private Sub cmdCancel_Click()
    Unload frmSetup
End Sub

Private Sub cmdDASetup_Click()
```

```

    frmDASetup.Show 1
End Sub

Private Sub cmdDefaults_Click()
    NewSetup = Default
    Call Init
End Sub

Private Sub cmdDelete_Click()
    If lstHeaters.ListCount > 0 Then
        Dim heater As Integer
        Dim Index As Integer
        '* assign the deleted heater to the text box
        '* so it could be quickly re-added
        Index = lstHeaters.ListIndex
        txtHeaters = lstHeaters.List(Index)
        heater = Val(lstHeaters.List(Index))
        '* remove heater from setup array
        NewSetup.Heaters.List(heater) = False
        '* decrement number of active heaters
        NewSetup.Heaters.Num = NewSetup.Heaters.Num - 1
        '* remove the heater from the list box
        lstHeaters.RemoveItem Index
        If Index < NewSetup.Heaters.Num Then
            lstHeaters.ListIndex = Index
        ElseIf NewSetup.Heaters.Num > 0 Then
            lstHeaters.ListIndex = Index - 1
        End If
    End If
End Sub

Private Sub cmdLoad_Click()
    Dim SetupFileNumber As Integer
    Dim SetupFileName As Integer
    '* set up properties of file box
    frmCalFile!cmn1.DefaultExt = ".CFG"
    frmCalFile!cmn1.DialogTitle = "Load Setup File"
    frmCalFile!cmn1.FILTER = "Setup Files (*.CFG)|*.CFG|All Files (*.*)|*.*"
    frmCalFile!cmn1.Flags = cd1OFNHideReadOnly + cd1OFNPathMustExist + cd1OFNFileMustExist
    On Error Resume Next
    frmCalFile!cmn1.ShowOpen
    If Not Err.Number = cd1Cancel Then
        '* return the file name from the text box
        NewSetup.FileName = frmCalFile!cmn1.FileName
        '* find free file number
        SetupFileNumber = FreeFile
        '* open data file for input
        Open NewSetup.FileName For Input As #SetupFileNumber
        '* call the subroutine that reads the data
        Call ReadCalSetup(NewSetup, SetupFileNumber)
        '* close the setup file
        Close SetupFileNumber
        '* re-initialize the form
        Call Init
    End If
End Sub

Private Sub cmdOK_Click()
    '* Look for parameters that cause problems.
    If NewSetup.Heaters.Range(0) > NewSetup.Heaters.Range(1) Then
        MsgBox "Number of Last Heater" + CRLF _
            + "is higher than First Heater."
    Else
        '* if all the parameters are ok, then
        '* setup i sok.
    End If
End Sub

```

```

        SetupData = NewSetup
        Call SetADCPParams
        Call SetADC
        SetupDone = True
        Unload frmSetup
    End If
End Sub

Private Sub cmdSave_Click()
    Dim SetupFileNumber As Integer
    Dim SetupFileName As String
    '* Initialize CommonDialogBox
    frmCalFile!cmn1.DefaultExt = ".CFG"
    frmCalFile!cmn1.DialogTitle = "Save Setup"
    frmCalFile!cmn1.FILTER = "Setup Files (*.CFG)|*.CFG|All Files (*.*)|*.*"
    frmCalFile!cmn1.Flags = cd1OFNHideReadOnly + cd1OFNOverwritePrompt + cd1OFNPathMustExist
    On Error Resume Next
    frmCalFile!cmn1.ShowSave
    If Not Err.Number = cdlCancel Then
        '* return a file name from dialog box
        NewSetup.FileName = frmCalFile!cmn1.FileName
        '* find a free file number
        SetupFileNumber = FreeFile
        '* open setup file for output
        Open NewSetup.FileName For Output As #SetupFileNumber
        '* call the subroutine that writes the data
        Call WriteCalSetup(NewSetup, SetupFileNumber)
        '* close the output file
        Close SetupFileNumber
    End If
End Sub

Private Sub Command1_Click()
End Sub

Private Sub Form_Load()
    '* Got rid of gain list - gain of 1 works for
    '* all situations
    For i = 0 To GainList.Num - 1
        cmbGain.AddItem GainList.Text(i), i
    Next i

    '*****
    '* Define Default Setup Values
    '*****
    Default.Vthresh = 300
    Default.Slew = 100
    Default.Vstep = 3
    Default.Vmax = 10
    Default.Rep = 1
    Default.numscans = 20
    Default.ScanRate = 1000
    Default.Gainindex = 0
    Default.Heaters.method = ALLACTIVE
    Default.Heaters.Num = 0
    Default.Comments = ""

    If SetupDone = True Then
        NewSetup = SetupData
    Else
        NewSetup = Default
    End If
    '* Initialize Form Variables
    Call Init

```

End Sub

```
Sub Init()
    '* several of the following lines were commented
    '* out when calibration routine was changed to
    '* bisection routine.
    txtVThresh.text = Str$(NewSetup.Vthresh)
    'txtSlew.Text = Str$(NewSetup.Slew)
    txtVStep.text = Str$(NewSetup.Vstep)
    'txtVmin.Text = Str$(NewSetup.Vmin)
    txtVMax.text = Str$(NewSetup.Vmax)
    'txtRep.Text = Str$(NewSetup.Rep)
    txtNumScans.text = Str$(NewSetup.numscans)
    txtScanRate.text = Str$(NewSetup.ScanRate)
    'commented out because cmbgain had no values in list
    'cmbGain.ListIndex = NewSetup.Gainindex
    txtADCOffset.text = Str(NewSetup.ADCOffset)

    txtRange(0) = NewSetup.Heaters.Range(0)
    txtRange(1) = NewSetup.Heaters.Range(1)
    Do While lstHeaters.ListCount > 0
        lstHeaters.RemoveItem 0
    Loop

    For i = 0 To 159
        If NewSetup.Heaters.List(i) = True Then
            lstHeaters.AddItem Str$(i)
        End If
    Next i
    If lstHeaters.ListCount > 0 Then
        lstHeaters.ListIndex = 0
    End If

    If NewSetup.Heaters.method = ALLACTIVE Then
        Call AllEnable
    ElseIf NewSetup.Heaters.method = RANGEACTIVE Then
        Call RangeEnable
    ElseIf NewSetup.Heaters.method = SPECIFIC Then
        Call SpecificEnable
    End If

    txtComments = NewSetup.Comments
    txtHeaters = ""

```

End Sub

```
Sub AllEnable()
    NewSetup.Heaters.method = ALLACTIVE
    txtRange(0).Enabled = False
    txtRange(1).Enabled = False
    txtHeaters.Enabled = False
    lstHeaters.Enabled = False
    cmdAdd.Enabled = False
    cmdDelete.Enabled = False
    optMethod(0).Value = True
    optMethod(1).Value = False
    optMethod(2).Value = False
End Sub

Sub RangeEnable()
    NewSetup.Heaters.method = RANGEACTIVE
    txtRange(0).Enabled = True
    txtRange(1).Enabled = True
    txtHeaters.Enabled = False
    lstHeaters.Enabled = False
    cmdAdd.Enabled = False
    cmdDelete.Enabled = False

```



```

    optMethod(0).Value = False
    optMethod(1).Value = True
    optMethod(2).Value = False

End Sub

Sub SpecificEnable()
    NewSetup.Heaters.method = SPECIFIC
    txtRange(0).Enabled = False
    txtRange(1).Enabled = False
    txtHeaters.Enabled = True
    lstHeaters.Enabled = True
    cmdAdd.Enabled = True
    cmdDelete.Enabled = True
    optMethod(0).Value = False
    optMethod(1).Value = False
    optMethod(2).Value = True

End Sub

Sub SetADCParams()
    SetupData.Gainindex = 0
    SetupData.ADChwtype = 1
    '* set the duration variable for use in the
    '* SetADC subroutine
    SetupData.Duration = SetupData.numscans / SetupData.ScanRate
    '* let samprate = scanrate to preserve
    '* compatibility with other program
    SetupData.SampRate = SetupData.ScanRate
    SetupData.Trigsrc = 0
    ADC.trigger = MANUALTRIG
    ADC.Freq = SetupData.ScanRate
    ADC.numscans = SetupData.numscans

    '* I don't want to turn on all the ADC.Channels
    '* right now.
    'If SetupData.Heaters.method = ALLACTIVE Then
    '    For i = 0 To 159
    '        ADC.Channels(i) = True
    '    Next i
    'ElseIf SetupData.Heaters.method = RANGEACTIVE Then
    '    For i = 0 To 159
    '        ADC.Channels(i) = False
    '    Next i
    '
    '    For i = SetupData.Heaters.Range(0) _
    '        To SetupData.Heaters.Range(1)
    '        ADC.Channels(i) = True
    '    Next i
    '
    ' ElseIf SetupData.Heaters.method = SPECIFIC Then
    '    For i = 0 To 159
    '        ADC.Channels(i) = False
    '    Next i
    '    For i = 0 To SetupData.Heaters.Num - 1
    '        ADC.Channels(lstHeaters.List(i)) = True
    '    Next i
    ' End If

End Sub

Private Sub Text1_Change()

End Sub

Private Sub optMethod_Click(Index As Integer)

```

```

    If Index = 0 Then
        Call AllEnable
    ElseIf Index = 1 Then
        Call RangeEnable
    ElseIf Index = 2 Then
        Call SpecificEnable
    End If

End Sub

Private Sub txtADCOffset_Change()
    NewSetup.ADCOffset = Val(txtADCOffset.text)

End Sub

Private Sub txtComments_Change()
    NewSetup.Comments = txtComments.text
End Sub

Private Sub txtHeaters_Change()
    Static Value
    Dim NewValue
    NewValue = Val(txtHeaters.text)
    If NewValue > -1 And NewValue < 160 Then
        Value = NewValue
    Else
        txtHeaters.text = Str$(Value)
    End If

End Sub

Private Sub txtHeaters_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        Call cmdAdd_Click
    End If

End Sub

Private Sub txtNumScans_Change()
    NewSetup.numscans = Val(txtNumScans.text)
End Sub

Private Sub txtNumScans_LostFocus()
    txtNumScans.text = Str$(NewSetup.numscans)
End Sub

Private Sub txtRange_Change(Index As Integer)
    Dim limit As Integer
    '* heater range is specified from 0 to 159
    limit = Val(txtRange(Index))
    If limit > -1 And limit < 160 Then
        NewSetup.Heaters.Range(Index) = limit
    End If

End Sub

Private Sub txtRange_LostFocus(Index As Integer)
    txtRange(Index).text = Str$(NewSetup.Heaters.Range(Index))

End Sub

Private Sub txtRep_Change()
    NewSetup.Rep = Val(txtRep.text)
End Sub

Private Sub txtRep_LostFocus()

```

```

        txtRep.text = Str$(NewSetup.Rep)
    End Sub

    Private Sub txtSampRate_Change()
        NewSetup.ScanRate = Val(txtScanRate.text)
    End Sub

    Private Sub txtScanRate_Change()
        NewSetup.ScanRate = Val(txtScanRate.text)
    End Sub

    Private Sub txtScanRate_LostFocus()
        txtScanRate.text = Str$(NewSetup.ScanRate)
    End Sub

    Private Sub txtSlew_Change()
        NewSetup.Slew = Val(txtSlew.text)
    End Sub

    Private Sub txtSlew_LostFocus()
        txtSlew.text = Str$(NewSetup.Slew)
    End Sub

    Private Sub txtTemp_Change()
        NewSetup.temp = Val(txtTemp.text)
    End Sub

    Private Sub txtTemp_LostFocus()
        txtTemp.text = Str$(NewSetup.temp)
    End Sub

    Private Sub txtVMax_Change()
        Dim newval As Double
        newval = Val(txtVMax.text)
        If newval <= 10000 And newval > 0 Then
            NewSetup.Vmax = newval
        End If
    End Sub

    Private Sub txtVMax_LostFocus()
        txtVMax.text = Str$(NewSetup.Vmax)
    End Sub

    Private Sub txtVmin_Change()
        Dim newval
        newval = Val(txtVmin.text)
        If newval >= 0 And newval < 10000 Then
            NewSetup.Vmin = Val(txtVmin.text)
        End If
    End Sub

    Private Sub txtVmin_LostFocus()
        txtVmin.text = Str$(NewSetup.Vmin)
    End Sub

    Private Sub txtVStep_Change()
        NewSetup.Vstep = Val(txtVStep.text)
    End Sub

    Private Sub txtVStep_LostFocus()
        txtVStep.text = Str$(NewSetup.Vstep)
    End Sub

```

```

Private Sub txtStepRate_Change()

End Sub

Private Sub txtVThresh_Change()
    NewSetup.Vthresh = Val(txtVThresh.text)
End Sub

Private Sub txtVThresh_LostFocus()
    txtVThresh.text = Str$(NewSetup.Vthresh)
End Sub

```

#### **G.2.4. Listing of FRMMAIN.FRM**

```

Option Explicit
*****
' declare variables for the calibration timer routine
*****
' keeps track of the element in the heaterlist array
' that is being tested.
Dim HeaterIndex As Integer
' keep track of the number of repetitions
Dim RepIndex As Integer
' contains a list of the heaters to be tested
Dim HeaterList(160) As Integer
' number of heaters to scan through
Dim NumHeaters
' hrs, min, seconds since beginning of test
Dim TestTime As Long
' number of seconds since midnight at start of test
' This will cause a confused test time if tests run
' past midnight, but it isn't a critical variable.
Dim StartTime As Long
' sum of Vcmd values from each repetition
Dim VcmdSum As Double
' number of Vcmd values that are summed to
' get the average that is saved in the calibration
' array
Dim NumVals As Integer
' generic variable used in averaging
Dim Sum As Double
'Vcmd stores the value of Vcmd to be output
' to the controller board
Dim Vcmd As Double
' VHeater contains the average of the scans of
' the heater output voltage
Dim VHeater As Double
' Vindex is used in the bisection routine to tell which
' of the three voltages is being applied
Dim Vindex As Integer
' array of three vcmd values that are used in subBisect
Dim V(3) As Double

Sub ADCDone()
    ' ADCDone doesn't do anything but call another subroutine because there is
    ' a subroutine in the ADC module that calls ADCDone to signal the end of
    ' data acquisition, but I initially used the subroutine StopCal for this
    ' same purpose before I started using that subroutine. I THINK that's
    ' what happened, anyway!

    Call StopCal
End Sub

```

```

Sub CalError()
    txtErrors.text = _
        "Calibration failed heater # " _
        + Str$(HeaterList(HeaterIndex)) _
        + CRLF + txtErrors.text
    Cal.Vcmd(HeaterList(HeaterIndex)) = 0
NextChannel

End Sub

Sub NewPoint(exceeded() As Integer)
    Dim DeltaV As Double
    Dim i As Integer
    '* calculate the new calibration datapoint to take
    DeltaV = V(2) - V(1)
    If Not exceeded(0) And exceeded(1) Then
        V(2) = V(1)
        exceeded(2) = exceeded(1)
        V(1) = V(0) + (V(2) - V(0)) / 2
    ElseIf Not exceeded(1) And exceeded(2) Then
        V(0) = V(1)
        exceeded(0) = exceeded(1)
        V(1) = V(0) + (V(2) - V(0)) / 2
    ElseIf Not exceeded(0) And Not exceeded(1) And Not exceeded(2) Then
        For i = 0 To 2
            V(i) = V(i) + DeltaV
        Next i
        Vindex = 0
    ElseIf exceeded(0) And exceeded(1) And exceeded(2) Then
        For i = 0 To 2
            V(i) = V(i) - DeltaV
        Next i
        Vindex = 0
    End If

    '* check if the last heater has been sampled
    '* check to see if DeltaV has become small enough to accept
    '* calibration point
    If DeltaV <= SetupData.Vmax Then
        SavePoint
    Else
        '* Otherwise, set the heater for the correct
        '* value
        Call SetHeater(HeaterList(HeaterIndex), V(1))
        '* set ADC sampling for this channel also
        ADC.Channels(HeaterList(HeaterIndex)) = True
    End If

    '* Check to see if newly-assigned Vcmd exceeds the limits
    If V(0) < 0 Or V(2) > 10000 Then
        CalError
        '* Set the returned Vcmd to zero and display error message
        '* go to the next channel
    End If
    '* set Vcmd to V(1)

End Sub

Sub SavePoint()
    '* call subroutine to save the value of Vcmd
    '* and change to next heater number
    '* Swap to Old Values
    Cal.Vcmd(HeaterList(HeaterIndex)) = Int(V(1))
    lblOldHeaterNum.Caption = lblHeaterNum.Caption
    lblOldVcmd.Caption = Str(Cal.Vcmd(HeaterList(HeaterIndex)))

```

```

        NextChannel

End Sub
Sub NextChannel()
    '* set Vcmd output to zero
    Call SetHeater(HeaterList(HeaterIndex), 0)
    '* turn off sampling on this channel
    ADC.Channels(HeaterList(HeaterIndex)) = False
    '* increment the heater index
    HeaterIndex = HeaterIndex + 1
    '* reset the index to zero
    Vindex = 0
    V(0) = 0
    V(1) = 5000
    V(2) = 10000
    '* check to see if we've sampled the last heater
    If HeaterIndex >= NumHeaters Then
        '* set current heater Vcmd to zero
        Call SetHeater(HeaterList(HeaterIndex), 0)
        '* call the CalStop routine
        Call StopCal
    Else
        Call SetHeater(HeaterList(HeaterIndex), V(0))
    End If

End Sub

Sub StopCal()
    '* This subroutine executes after the calibration is complete. It save the
    '* calibration data and shuts everything off.

    Dim FileNumber As Integer
    '* disable timer
    tmr1.Enabled = False

    '* toggle command buttons
    cmdStart.Enabled = True
    txtFileName.Enabled = True
    cmdBrowse.Enabled = True
    txtComments.Enabled = True
    cmdSetup.Enabled = True
    cmdStop.Enabled = False

    '* set Vcmd = 0
    Call SetHeater(HeaterList(HeaterIndex), 0)
    '* turn off sampling of that channel
    ADC.Channels(HeaterList(HeaterIndex)) = False

    '* Save Vcmd File
    '* find a free filename
    FileNumber = FreeFile
    '* open file using textbox name, for output
    Open txtFileName.text For Output As FileNumber
    '* write setup data
    Call WriteCalSetup(SetupData, FileNumber)
    '* write Vcmd Data
    Call WriteCal(Cal, FileNumber)
    Close FileNumber
    '* Change Status Indicator
    lblStatus.Caption = "Inactive"
    lblStatus.ForeColor = RGB(255, 0, 0)

End Sub
Sub subBisect()
    Static exceeded(3) As Integer

```

```

Dim i As Integer, OverThresh As Integer
'* On the first sample, Vcmd should have already
'* been set to it's first value in cmdStart.

'* Turn on ADC array for desired heater to sample
'* All other ADC values should have been
'* set in setup
ADC.Channels(HeaterList(HeaterIndex)) = True

'* Take Data
'* I want ADCGetData to return mV readings
ADCGetData
'* Average scan values
Sum = 0
For i = 0 To ADC.numscans - 1
    Sum = Sum + ADC.Buffer(0, i)
Next i
'* Vheater is given in mV
VHeater = Sum / ADC.numscans

'* Check heater voltage for Threshold voltage
If VHeater > SetupData.Vthresh Then
    OverThresh = True
Else
    OverThresh = False
End If

'* check if all three vcmd values have been tested. Then decide what
'* three values of vcmd to test next.
If Vindex < 3 Then
    '* assign threshold value to indexed array
    exceeded(Vindex) = OverThresh
    '* increment the Vcmd index
    Vindex = Vindex + 1
    '* set vcmd to V(vindex)
    Call SetHeater(HeaterList(HeaterIndex), V(Vindex))
    '* set ADC sampling for this channel also
Else
    exceeded(1) = OverThresh
End If
If Vindex = 3 Then
    Call NewPoint(exceeded)
End If
UpdateWindow
End Sub

Sub subExit()
    If frmConfirm.yesno("Exit Heater Calibration?") = True Then
        Unload frmMain
    End If
End Sub

Sub UpdateWindow()
    '* Update the values displayed in the window control.
    '* Update test time
    TestTime = CLng(Timer) - StartTime
    lblTestTime.Caption = Str$(TestTime)
    lblHeaterNum.Caption = Str$(HeaterList(HeaterIndex))
    lblVcmd.Caption = Str$(Vcmd)
    lblReps.Caption = Str$(RepIndex)
End Sub

Private Sub cmdBrowse_Click()
    '* Load a common-dialog window to browse calibration output files
    frmCalFile!cmn1.DefaultExt = ".CAL"
    frmCalFile!cmn1.DialogTitle = "Data File Name"

```

```

frmCalFile!cmn1.FILTER = "Calibration Files (*.CAL)|*.CAL|All Files (*.*)|*.*"
frmCalFile!cmn1.Flags = cd1OFNHideReadOnly + cd1OFNOverwritePrompt + cd1OFNPathMustExist
On Error Resume Next
frmCalFile!cmn1.ShowSave
If Not Err.Number = cd1Cancel Then
    txtFileName = frmCalFile!cmn1.FileName
    Call txtFileName_Change
End If

End Sub

Private Sub cmdExit_Click()
    '* call the subroutine to prompt the user to confirm exit
    Call subExit
End Sub

Private Sub cmdSetup_Click()
    '* Load the setup form, to set parameters in the program
    frmSetup.Show 1

    '* enable the start button once setup is complete. Thus the user is
    '* required to go to setup and press "OK" before any calibration
    '* can be performed.

    '* Also checks to see if a filename has been entered in the filename box

    If SetupDone = True _
        And Not ADC.FileName = "" Then
        cmdStart.Enabled = True
    End If

End Sub

Private Sub cmdStart_Click()
    '* Begin the calibration

    '* Disable everything we don't want user to
    '* mess with while collecting data.
    Dim i As Integer
    cmdStart.Enabled = False
    txtFileName.Enabled = False
    cmdBrowse.Enabled = False
    txtComments.Enabled = False
    cmdSetup.Enabled = False
    cmdStop.Enabled = True

    '* Turn on the status indicator
    lblStatus.Caption = "Calibrating"
    lblStatus.ForeColor = RGB(0, 255, 0)

    '* set the timer interval - mult. by 1000 to convert to ms
    tmr1.interval = 1000 / SetupData.Vstep
    '* enable the timer
    tmr1.Enabled = True

    '* initialize variables for the calibration routine
    V(0) = 0
    V(1) = 5000
    V(2) = 10000
    '* initially set Vindex to zero
    Vindex = 0
    HeaterIndex = 0
    VcmdSum = 0
    NumVals = 0
    txtErrors.text = ""
    '* initialize the start time, update after each
    '* calibration
    StartTime = Timer
    TestTime = 0

    '* go through and initially make sure sampling

```



```

    '* is turned off on all the channels. It accidentally
    '* got turned on in SetADC. Gotta fix that!
    For i = 0 To ADCLASTCHANNEL
        ADC.Channels(i) = False
    Next i

    '* form heaterlist array to make program simpler
    '* The number of valid elements in this array
    '* should be the same as setupdata.heaters.num
    If SetupData.Heaters.method = SPECIFIC Then
        For i = 0 To 159
            If SetupData.Heaters.List(i) = True Then
                HeaterList(HeaterIndex) = i
                HeaterIndex = HeaterIndex + 1
            End If
        Next i
        NumHeaters = SetupData.Heaters.Num
    ElseIf SetupData.Heaters.method = RANGEACTIVE Then
        For i = SetupData.Heaters.Range(0) To SetupData.Heaters.Range(1)
            HeaterList(HeaterIndex) = i
            HeaterIndex = HeaterIndex + 1
        Next i
        NumHeaters = SetupData.Heaters.Range(1) _
            - SetupData.Heaters.Range(0) + 1
    ElseIf SetupData.Heaters.method = ALLACTIVE Then
        For i = 0 To 159
            HeaterIndex = i
            HeaterList(HeaterIndex) = i
        Next i
        NumHeaters = 96
    End If

    '* reset heaterindex to zero
    HeaterIndex = 0
    '* set vcmd to V(vindex)
    Call SetHeater(HeaterList(HeaterIndex), V(Vindex))

    '* update the values displayed in the main window
    Call UpdateWindow

End Sub
Private Sub cmdStop_Click()
    '* call subroutine to stop the calibration
    Call StopCal
End Sub

Private Sub Form_Load()

    '* initialize control values
    cmdStop.Enabled = False
    cmdStart.Enabled = False
    CRLF = Chr$(13) + Chr$(10)
    txtTemp.text = ""
    txtErrors.text = ""
    txtFileName.text = ""
    txtComments.text = ""
    lblStatus.Caption = "Inactive"
    lblStatus.ForeColor = RGB(255, 0, 0)
    lblTestTime.Caption = ""
    lblHeaterNum.Caption = ""
    lblVcmd.Caption = ""
    lblReps.Caption = ""
    lblOldHeaterNum.Caption = ""
    lblOldVcmd.Caption = ""

    '* temporarily use to set hardware selections
    ADCHardwareType = ADCDAQBOOK
    DACHardwareType = DACCCB

```

```

    '* set daqbook parallel port protocol
    ADCPPP = ADC8BIT

    SetupData.ADChwtype = ADCDAQBOOK

    '* when program is initially loaded, setup has not been performed, so set
    '* this variable.
    SetupDone = False

    '* cause error to result if the file load/save
    '* form is canceled.
    frmCalFile!cmn1.CancelError = True
    '* initialize A/D variables
    ADCDefine
    Call DACInit

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If Not UnloadMode = vbFormCode Then
        Call subExit
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    DACClose
    ADCClose
    End
End Sub

Private Sub tmr1_Timer()
    subBisect
End Sub

Private Sub txtComments_Change()
    '* text in this text box is saved with the calibration file
    Cal.Comments = txtComments.text
End Sub

Private Sub txtFileName_Change()
    ADC.FileName = txtFileName
    If ADC.FileName = "" Then
        cmdStart.Enabled = False
    ElseIf SetupDone = True Then
        cmdStart.Enabled = True
    End If
End Sub

Private Sub txtTemp_Change()
    Cal.temp = Val(txtTemp.text)
End Sub

Private Sub txtTemp_LostFocus()
    txtTemp = Str$(Cal.temp)
End Sub

```

### G.3. LISTING OF GRAPH1.VBP

```

Form=FRMMAIN.FRM
Module=modShared; ..\CONTROL\MODSHARE.BAS
Module=modMain; ..\CONTROL\MODMAIN.BAS
Module=modADC; ..\TRIM\MODADC.BAS
Module=modDaqBook; ..\DAQBOOK.BAS
Form=..\CONTROL\FRMCALFI.FRM
Form=..\CONTROL\FRMADC.FRM
Module=modGraph; MODGRAPH.BAS
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.0#0; COMDLG16.OCX
Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.0#0; TABCTL16.OCX
Object={FAEEE763-117E-101B-8933-08002B2F4F5A}#1.0#0; DBLIST16.OCX
Object={00028C01-0000-0000-0000-000000000046}#1.0#0; DBGRID16.OCX
Reference=*\G{BEF6E001-A874-101A-8BBA-00AA00300CAB}#1.0#0#C:\WINDOWS\SYSTEM\OC25.DLL#Standard OLE
Types
Reference=*\G{00025E01-0000-0000-C000-000000000046}#2.5#0#C:\WINDOWS\SYSTEM\DAO2516.DLL#Microsoft
DAO 2.5 Object Library
ProjWinSize=19,314,251,413
ProjWinShow=2
IconForm="frmMain"
ExeName="GRAPH1.EXE"
Path="F:\BIN"
Name="Project1"
HelpContextID="0"
StartMode=0
VersionCompatible="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="University of Denver"

```

#### G.3.1. Listing of MODGRPAH.BAS

```

Attribute VB_Name = "modGraph"
'* These functions and constants are necessary only to keep the compiler
'* from complaining, without including cbw.bas. Not even necessary when running
'* without a full compile.

Public Const FIRSTPORTA = 0
Public Const FIRSTPORTB = 0
Public Const firstportcl = 0
Public Const FIRSTPORTCH = 0
Public Const DIGITALOUT = 0
Public Const digitalin = 0

Function cbdabin(a, b, c, d)
    '* dummy function
End Function

Function cbdbitout(a, b, c, d)
    '* dummy function
End Function

Function cbdconfigport(a, b, c)
    '* dummy function
End Function

Function cbdin(a, b, c)
    '* dummy function

```

```

End Function

Function cbdout(a, b, c)
    '* dummy function
End Function

```

### **G.3.2. Listing of FRMMAIN.FRM**

```

Option Explicit
Dim OutData As Double
Dim NumScans As Integer
Dim HeaterColor As Long
Dim SetupIn As SetupType
Dim x(160) As Integer, y(160) As Integer
Dim Data(100, 2000) As Double
Dim Colors(100, 2000) As Long

Dim MaxPower As Double

Sub ADCDone()

    '* Dummy subroutine so it will compile

End Sub

Sub Animate()
    Dim i
    Dim j
    Dim Time As Integer, TimeStep As Double
    Dim TimePrint As Integer

    Time = 0
    '* TimeStep is in ms
    TimeStep = 1 / SetupIn.SampRate * 1000
    TimePrint = 0

    For i = 0 To NumScans - 1

        For j = 0 To SetupIn.heaters.Num - 1
            Line (x(j), y(j))-(x(j) + 230, y(j) + 230), Colors(j, i), BF
        Next j
        '* Time is in ms
        Time = TimeStep * i
        TimePrint = TimePrint + 1
        If TimePrint = 10 Then
            txtTime.text = Str(Time)
            TimePrint = 0
            txtTime.Refresh
        End If

    Next i

End Sub

Function FalseColor(OutData As Double) As Long
    Dim R As Integer, G As Integer, b As Integer, temp As Integer
    temp = OutData * 255 * 5 / MaxPower
    If temp <= 255 Then
        R = 0
        G = 0
        b = temp
    ElseIf temp <= 256 * 2 Then
        R = 0
        G = -255 + temp
        b = 255
    ElseIf temp <= 256 * 3 Then
        R = 0

```

```

        G = 255
        b = 256 * 3 - temp
    ElseIf temp <= 256 * 4 Then
        R = temp - 256 * 3
        G = 255
        b = 0
    ElseIf temp <= 256 * 5 Then
        R = 255
        G = 256 * 5 - temp
        b = 0
    Else
        R = 255
        G = 255
        b = 255
    End If

    FalseColor = RGB(R, G, b)
End Function

Sub LoadBin()
    Dim DataPoint As Integer
    Dim ScanLine As String, Sep As String
    Dim j As Integer
    Dim strDate As String
    Dim strTime As String
    Dim Comments As String
    Dim InputFileNumber As Integer
    Dim OutputFileNumber As Integer
    Dim TagFileNumber As Integer
    Dim lngDataPoint As Long
    Dim Formatted As String
    Dim heaters(160) As Integer
    Dim PercentComplete As Long, oldPC As Long
    Dim i As Integer, FullScale As Integer
    Dim heaterindex As Integer

    PlotScale

    InputFileNumber = FreeFile
    On Error Resume Next
    Open ChgExt(txtInputFile.text, ".TAG") For Input As InputFileNumber
    If Err.Number = 53 Then
        MsgBox (Err.Description)
        Close InputFileNumber
        Exit Sub
    End If
    Call ReadSetup(SetupIn, InputFileNumber)
    Close InputFileNumber
    Open txtInputFile.text For Binary As InputFileNumber Len = Len(DataPoint)
    NumScans = SetupIn.Duration * SetupIn.SampRate
    If frmMain!txtsteps.text > 0 Then
        If NumScans > Val(frmMain!txtsteps.text) Then NumScans = Val(frmMain!txtsteps.text)
    End If

    '* set the voltage scaling to the correct value
    If ADCList.Value(SetupIn.ADChwtype) = ADCDAQBOOK Then
        FullScale = 10
    ElseIf ADCList.Value(SetupIn.ADChwtype) = ADCCUSTOM Then
        FullScale = 12
    Else
        FullScale = 10
    End If
    '* in case we read old files that don't have the extra setup data
    End If

    '* the following code just converts all the other
    '* heater specification methods to the list method.
    '* This is just a work-around for the way this routine was
    '* originally written.

    Select Case SetupIn.heaters.method
        Case ALLACTIVE

```

```

        For i = 0 To ADCLASTCHANNEL
            SetupIn heaters.List(i) = i
            SetupIn heaters.Num = ADCLASTCHANNEL + 1
        Next i
    Case RANGEACTIVE
        SetupIn heaters.Num = SetupIn heaters.Range(1) _
            - SetupIn heaters.Range(0) + 1
        For i = 0 To ADCLASTCHANNEL
            If i >= SetupIn heaters.Range(0) _
                And i <= SetupIn heaters.Range(1) Then
                SetupIn heaters.List(i) = True
            Else
                SetupIn heaters.List(i) = False
            End If
        Next i
    End Select

    heaterindex = 0
    For i = 0 To 159
        If SetupIn heaters.List(i) = True Then
            heaters(heaterindex) = i
            x(heaterindex) = lblHeater(heaterindex).Left
            y(heaterindex) = lblHeater(heaterindex).Top
            heaterindex = heaterindex + 1
        End If
    Next i

    For i = 0 To NumScans - 1
        For j = 0 To SetupIn heaters.Num - 1
            '* input one data point
            Get InputFileNumber, , DataPoint
            lngDataPoint = DataPoint
            OutData = (CDBl(lngDataPoint) / 2 ^ 16 _
                * FullScale) _
                / 1
            If OutData < 0 Then OutData = OutData + FullScale
            '* Val(GainList.Text(SetupIn.Gainindex))
            Formatted = Format(OutData, "Scientific")
            Data(j, i) = OutData
            HeaterColor = FalseColor(OutData ^ 2)
            Colors(j, i) = HeaterColor
        Next j
    Next i

    '* close files
    Close InputFileNumber
End Sub

Sub PlotScale()
    Dim i As Integer
    For i = 0 To 14
        HeaterColor = FalseColor(i * MaxPower / 14)
        lblHeater(96 + i).BackColor = HeaterColor
        lblHeater(96 + i).Caption = Str(Int(i * MaxPower / 14))
    Next i
    frmMain.Refresh
End Sub

Sub ReColor()
    Dim i
    Dim j
    MaxPower = Val(txtMaxPower.text)
    If MaxPower = 0 Then MaxPower = 100
    PlotScale
    For i = 0 To NumScans - 1
        For j = 0 To SetupIn heaters.Num - 1
            HeaterColor = FalseColor(Data(j, i) ^ 2)
            Colors(j, i) = HeaterColor
        Next j
    Next i
End Sub

```

```

        Next j
    Next i
End Sub

Private Sub Text1_DblClick()
    '* Browse file names
End Sub

Private Sub cmdReScale_Click()
    ReColor
End Sub

Private Sub cmdRestart_Click()
    Animate
End Sub

Private Sub cmdStep_Click()
    Static i
    Static j
    Static Time As Integer, TimeStep As Double
    Static TimePrint As Integer

    '* TimeStep is in ms
    TimeStep = 1 / SetupIn.SampRate * 1000

    For i = i To i + Val(txtFrames)
        For j = 0 To SetupIn.heaters.Num - 1
            Line (x(j), y(j))-(x(j) + 230, y(j) + 230), Colors(j, i), BF
        Next j
        '* Time is in ms
        Time = TimeStep * i
        TimePrint = TimePrint + 1
        If TimePrint = 10 Then
            txtTime.text = Str(Time)
            TimePrint = 0
            txtTime.Refresh
        End If
    Next i

    If Time > SetupIn.Duration Then Time = 0
End Sub

Private Sub Command2_Click()
    LoadBin
End Sub

Private Sub Form_Load()
    ADCDefine
    MaxPower = 100
    txtMaxPower.text = "100"
    txtInputFile.text = ""
    txtTime.text = "0"

    lblHeater(111).BackColor = RGB(255, 255, 255)
    lblHeater(111).ForeColor = RGB(0, 0, 0)
    lblHeater(111).Caption = "OVER"
    PlotScale
End Sub

Private Sub txtInputFile_DblClick()

    Static FileName As String
    '* set up properties of file box
    frmCalFile!cmn1.FileName = FileName
    frmCalFile!cmn1.DefaultExt = ".BIN"
    frmCalFile!cmn1.DialogTitle = "Binary Input File"

```

```
frmCalFile!cmn1.FILTER = "Binary Data Files(*.BIN)|*.BIN|All Files(*.*)|*.*"  
frmCalFile!cmn1.Flags = cd1OFNHideReadOnly + cd1OFNPathMustExist + cd1OFNFileMustExist  
On Error Resume Next  
frmCalFile!cmn1.ShowOpen  
If Not Err.Number = cd1Cancel Then  
    frmMain!txtInputFile.text = frmCalFile!cmn1.FileName  
    FileName = frmCalFile!cmn1.FileName  
    'Call txtInputFile_Change  
End If  
  
End Sub
```





REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE April 1998		3. REPORT TYPE AND DATES COVERED Final Contractor Report
4. TITLE AND SUBTITLE Design, Construction, and Qualification of a Microscale Heater Array for Use in Boiling Heat Transfer			5. FUNDING NUMBERS  WU-962-24-00-00 NAG3-1609	
6. AUTHOR(S)  T.D. Rule, J. Kim, and T.S. Kalkur				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Washington State University School of Mechanical and Materials Engineering Pullman, Washington 99164-2920			8. PERFORMING ORGANIZATION REPORT NUMBER  E-11167	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191			10. SPONSORING/MONITORING AGENCY REPORT NUMBER  NASA CR-1998-207407	
11. SUPPLEMENTARY NOTES  T.D. Rule, Washington State University, School of Mechanical and Materials Engineering, Pullman, Washington 99164-2920; J. Kim, University of Denver, Department of Engineering, Denver, Colorado 80208; T.S. Kalkur, University of Colorado, Department of Electrical Engineering, Colorado Springs, Colorado 80933-7150. Project Manager, John McQuillen, Microgravity Science Division, NASA Lewis Research Center, organization code 6712, (216) 433-2876.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT  Unclassified - Unlimited Subject Category: 34  This publication is available from the NASA Center for AeroSpace Information, (301) 621-0390.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  Boiling heat transfer is an efficient means of heat transfer because a large amount of heat can be removed from a surface using a relatively small temperature difference between the surface and the bulk liquid. However, the mechanisms that govern boiling heat transfer are not well understood. Measurements of wall temperature and heat flux near the wall would add to the database of knowledge which is necessary to understand the mechanisms of nucleate boiling. A heater array has been developed which contains 96 heater elements within a 2.5 mm square area. The temperature of each heater element is held constant by an electronic control system similar to a hot-wire anemometer. The voltage that is being applied to each heater element can be measured and digitized using a high-speed A/D converter, and this digital information can be compiled into a series of heat-flux maps. Information for up to 10,000 heat flux maps can be obtained each second. The heater control system, the A/D system and the heater array construction are described in detail. Results are presented which show that this is an effective method of measuring the local heat flux during nucleate and transition boiling. Heat flux maps are obtained for pool boiling in FC-72 on a horizontal surface. Local heat flux variations are shown to be three to six times larger than variations in the spatially averaged heat flux.				
14. SUBJECT TERMS  Multiphase flow; Two phase flow; Boiling heat transfer; Microinstrumentation; Measure instruments			15. NUMBER OF PAGES 263	
			16. PRICE CODE A12	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	